Deep Neural Networks based on Gating Mechanism for Open-Domain Question Answering

Christian Alex Mayhua Tijera

Advisor: José Eduardo Ochoa Luna, Ph.D.

Examining Committee:
Camilo Thorne Ph.D. – Elsevier – Germany
Juan Antonio Lossio Ventura Ph.D. – University of Florida – USA
Edwin Villanueva Talavera Ph.D. – Pontificia Universidad Católica del Perú – Perú
Alex Cuadros Vargas Ph.D. – Universidad Católica San Pablo – Perú

Thesis submitted to the Department of Computer Science in partial fulfillment of the requirements for the degree of Master in Computer Science.

Universidad Católica San Pablo – UCSP
December of 2018 – Arequipa – Perú

To my wife Lizeth, for always encouraging me to overcome my own limits. To my child Drako, for inspiring me with his infinite curiosity. To my mother Julia, for supporting me all these years.

Acknowledgments

First of all I want to thank my family for encouraging me to follow my research path.

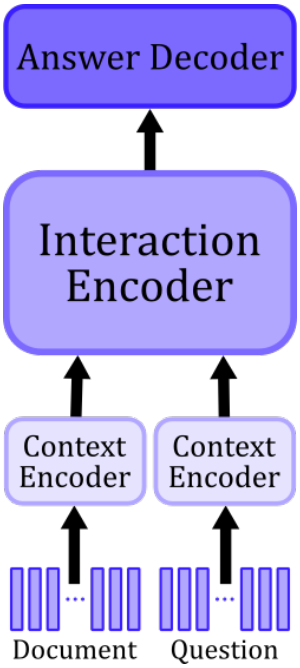
I would like to thank in a special way the National Council of Science, Technology and Technological Innovation (CONCYTEC) and the National Fund for Scientific, Technological development and Technological Innovation (FONDECYT-CIENCIACTIVA), which through the Management Agreement Nº 234-2015-FONDECYT, they have allowed the grant and financing of my studies of Master in Computer Science at the Universidad Católica San Pablo (UCSP).

I thank in a very special way my advisor Prof. José Eduardo Ochoa Luna, Ph.D. for had guided me in this thesis.

I thank all my professors for their teachings.

I thank in a special way the examining committee, for taking the time to read my thesis and give me good feedback.

Abstract



Nowadays, Question Answering is being addressed from a reading comprehension approach. Usually, Machine Comprehension models are powered by Deep Learning algorithms. Most related work faces the challenge by improving the Interaction Encoder, proposing several architectures strongly based on attention. In Contrast, few related work has focused on improving the Context Encoder. Thus, our work has explored in depth the Context Encoder. We propose a gating mechanism that controls the flow of information, from the Context Encoder towards Interaction Encoder. This gating mechanism is based on additional information computed previously. Our experiments has shown that our proposed model improved the performance of a competitive baseline model. Our single model reached 78.36% on F1 score and 69.1% on exact match metric, on the Stanford Question Answering benchmark.

Keywords: Machine Comprehension, Question Answering, Natural Language Processing, Deep Learning.

Contents

List of Tables

List of Figures

List of Acronyms

AD Answer Decoder

AI Artificial Intelligence

BiDAF Bi-Directional Attention Flow

CE Context Encoder

CoVe Context Vectors

CNN Convolutional Neural Network

DL Deep Learning

EM Exact Match

GloVe Global Vectors

GRU Gated Recurrent Unit

HoW history-of-word

IE Interaction Encoder

IR Information Retrieval

LSTM Long Short Term Memory

MC Machine Comprehension

MT Machine Translation

NER Named Entity Recognition

NLP Natural Language Processing

OOV out-of-vocab

POS Part-Of-Speech

QA Question Answering

RL Reinforcement learning

RNN Recurrent Neural Network

SAN Stochastic Answer Network

SQuAD Stanford Question Answering Dataset

TF Term Frequency

TREC Text Retrieval Conference

List of Symbols

Matrix Operations

$A \circ B$	Binary operation
$A + B$	Element-wise addition
$A \times B$	Element-wise product
$A \cdot B$	Matrix multiplication
$[A; B]$	Matrix concatenation by rows
$[A, B]$	Matrix concatenation by columns
$A^\top$	Matrix transpose

Activation Functions

f	Non-linear function
σ	Sigmoid function
$tanh$	Hyperbolic tangent function

Neural Networks

W_l	Weights matrix related to layer l
b_l	Bias related to layer l

Chapter 1

Introduction

Question Answering (QA) is a sophisticated form of Information Retrieval (IR) characterized by information needs that are expressed as natural language statements or questions (Kolomiyets and Moens, 2011).

The first developed QA models (Green et al., 1961; Woods, 1978) were essentially natural language interfaces for expert systems, whose common feature was that they had a core based on knowledge databases hand-written by experts on a specific domain, also known as Closed-Domain. In contrast, current QA models use text documents as a knowledge base and combine several Natural Language Processing (NLP) techniques. Often, the collection of documents are not specialized on a specific topic and are called Open-Domain (Rajpurkar et al., 2016).

Some of the early Artificial Intelligence (AI) systems included QA skills. The most important and famous system was ELIZA (Weizenbaum, 1966). It simulated a conversation with a psychologist. ELIZA was able to talk about any topic by using very simple rules that detected important words in the input. It was a very rudimentary model to answer questions, but it generated a series of chatbots that participated in the annual Loebner Prize[1].

In the late 90s, the annual Text Retrieval Conference (TREC)[2] included Open-Domain QA as one of its competition tracks (Dang et al., 2007). The challenge was to return a concise answer to a natural language question, given a large collection of text documents. TREC had a major impact on interest in QA and on the development of evaluation measures that compare models performance (Voorhees et al., 2005).

In 2011, the IBM's Watson won the quiz show Jeopardy![3]. Watson is a QA model that parses questions into different keywords and sentence fragments in order to find

[1]The Loebner Prize is an annual competition, whose format follows the standard established in the Turing test. http://www.aisb.org.uk/events/loebner-prize

[2]TREC is a series of workshops dedicated to a list of different IR research areas. https://trec.nist.gov

[3]Jeopardy! is a knowledge contest with questions on many topics. https://www.jeopardy.com

statistically related phrases (Ferrucci et al., 2010). Watson's main innovation was not in the creation of a new algorithm but rather its ability to quickly execute hundreds of NLP algorithms simultaneously. The more algorithms that find the same answer independently the more likely Watson is to be correct.

The current QA trend begins in 2013. Since, Machine Comprehension (MC) becomes the most promising way to perform QA, powered by the creation of several benchmark datasets (Richardson et al., 2013; Hermann et al., 2015; Hill et al., 2015; Rajpurkar et al., 2016). The aim of these benchmarks were focused on answering questions, in order to evaluate the ability of a machine to understand text. In these benchmarks, a piece of text called the Document is presented to the model. Then, the model is expected to be able to answer any Questions about the related text.

Nowadays, QA models powered by Deep Learning (DL) have become the state-of-the-art (Hu et al., 2018; Wang et al., 2017; Liu et al., 2018). These models are basically composed by three modules: The first one is the Context Encoder (CE) that is responsible for encoding the words of the document and the question according to their surrounding words. The Interaction Encoder (IE) encodes the interaction between the document and the question. Finally, the Answer Decoder (AD) extracts the answer to the question based on the previous encoding. The work presented in this thesis follows the same approach.

1.1 Motivation and Context

Recent DL emergence is based on three pillars: the improvement of algorithms related to neural networks, a large amount of data and a greater processing capacity. In this context, many research fields have been approached with DL which also caused the development of some QA datasets in recent years. These datasets have different styles, such as: choose from multiple options (Richardson et al., 2013), fill in the blank space (Hermann et al., 2015), and extract a piece of text (Rajpurkar et al., 2016).

Thus, a lot of recent research follows a DL approach (Huang et al., 2018; Hu et al., 2018; Pan et al., 2017; Shen et al., 2017a; Wang et al., 2017; Seo et al., 2017). In fact, one of these models is currently leading the state-of-the-art (Yu et al., 2018); being about 1% below human performance (Rajpurkar et al., 2016). Most of these models follow an Encoder-Decoder scheme. Due to the nature of the task, it is necessary to encode the word sequences, in order to extract an answer from these sequences.

In this context, our main motivation is to contribute to the development of smarter AI, through the building of QA models that will be able to answer a wide variety of questions. This is an important step in order to propitiate a most natural human-computer interaction.

1.2 Problem Statement

Artificial Intelligence (AI) is getting smarter every day and QA is a fundamental task for this aim. However, current QA models are not able to answer all the questions yet. Although great advances are being made by Deep Learning, the QA models still do not outperform the accuracy achieved by humans.

Most Research works have focused on Interaction Encoder (IE). However, the Context Encoder (CE) is an important component in the baseline. The CE is responsible for encoding word sequence in a contextual framework. Usually, CE processes Document and Question sequences independently. Since, current Machine Comprehension (MC) benchmarks do not have too much data (Yu et al., 2018), CEs do not be able to learn a strong language model. Thus, it is necessary to improve the CE architectures.

1.3 Objectives

Propose a Deep Neural Network model for improving a competitive baseline model in Open-Domain QA for English language, adding a simple structure called gating mechanism into Context Encoder.

1.3.1 Specific Objectives

- Improve the Context Encoder by testing several setups between gating mechanisms and Recurrent Neural Network (RNN).
- Improve the Context Encoder by applying multi-branch processing, generating multiple contextual representations.
- Improve the Context Encoder by proposing a fusing mechanism, in order to merge multiple contextual representations.
- Perform an ablation study about proposed components, in order to measure the impact of each Context Encoder component.

1.4 Contributions

Our contributions are focused on improving the CE, without adding external features to the benchmark. In such a way, We experimented with several CE architectures, achieving to improve the performance of the competitive baseline model proposed by Seo et al. (2017).

Our main contribution was to add a second parallel branch of processing, into the CE. This second branch strengths the model, adding an alternative contextual representation. Our multi-branch CE generated a publication for the 17th Mexican International Conference on Artificial Intelligence—MICAI 2018[4]—that was held on October 22-27 in Guadalajara, Mexico (Tijera and Ochoa-Luna, 2018).

Another important contribution is the fusing of branches with a gating mechanism. This gate allowed us to control, smartly, the flow of information of each branch towards IE. We perform an conscientious ablation analysis about features included in the gate. Finally, we did a depth analysis about Highway layers impact across the model.

1.5 Thesis Outline

This thesis is organized in 6 chapters for a better understanding and development. Chapter 2 presents a background to understand the basic concepts behind our proposal. Chapter 3 presents a review of DL-based QA models. Chapter 4 presents the thesis proposal based also on DL.

Chapter 5 presents our experiments, implementation details, evaluation methodology and results obtained with the proposed model. In addition, a comprehensive analysis of results with respect to the metrics used is also described. Finally, Chapter 6 presents future work and conclusions of this thesis.

[4] www.MICAI.org

Chapter 2

Background

This chapter presents essential background concepts to support the proposal. First, a brief definition of current way to perform Question Answering (QA), called Machine Comprehension (MC). Then, a description of general pipeline for performing MC. Next, we present Deep Learning (DL)-based architectures included in our model, these are ordered by its influence in pipeline components.

2.1 Machine Comprehension (MC)

Reading comprehension is defined as the ability to read text, process it, understand its meaning and then be able to answer any questions about it (Grabe, 2009). When machines perform this task it is called Machine Comprehension (MC). Although this definition may seem simple, this is a challenging task for machines. The next subsection analyzes several kinds of benchmarks for MC tasks which are available until this moment.

2.1.1 Machine Comprehension Benchmarks

The significant advance on MC has largely benefited from the availability of large-scale datasets. Existing datasets can be classified into three categories according to how they were labeled (Wang and Jiang, 2017).

The first category comprises datasets labeled by humans which are always in high quality (Richardson et al., 2013; Berant et al., 2014; Yang et al., 2015), but are too small for training modern data-intensive models. MCTest (Richardson et al., 2013) is one of the famous and high quality datasets. There are 660 fictional stories and 4 multiple choice questions per story contained in it. The labels are all made by humans. The objective of this type of dataset is to choose an answer between a set of alternatives.

Nikola Tesla	
In 1870, Tesla moved to Karlovac, to **attend school at the Higher Real Gymnasium**, where he was profoundly influenced by a math teacher **Martin Sekulic**. The classes were held in **German**, as it was a school within the Austro-Hungarian Military Frontier. Tesla was able to perform integral calculus in his head, which prompted his teachers to believe that he was cheating. He finished a four-year term in three years, graduating in 1873.	
In what language were the classes given?	**German**
Who was Tesla's main influence in Karlovac?	**Martin Sekulic**
Why did Tesla go to Karlovac?	**attend school at the Higher Real Gymnasium**

Table 2.1: Question-Answer pairs extracted from SQuAD v1.1. Each answer is a text segment of the paragraph.

The second category comprises datasets automatically generated from natural occurring data. These can be very large (Hill et al., 2015; Hermann et al., 2015; Onishi et al., 2016; Paperno et al., 2016) and allow the training of more expressive models. However, they are in cloze style. The goal is to predict the missing word in a document. Moreover, a thorough examination of the CNN/Daily Mail dataset (Hermann et al., 2015) has shown that this requires less reasoning than previously thought (Chen et al., 2016).

The third category involves extraction (Rajpurkar et al., 2016; Joshi et al., 2017) where the answer to each question is a segment of text from the corresponding document. In this sense, Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) provides large and high-quality questions. The answers in SQuAD often can have much longer phrases which is more challenging than cloze style datasets. We present an example extracted from SQuAD v1.1 (Rajpurkar et al., 2016) in Table 2.1.

SQuAD (Rajpurkar et al., 2016) retains a diverse set of answers and requires different forms of logical reasoning, including multi-sentence reasoning, such as: question "Why did Tesla go to Karlovac?", presented in Table 2.1. Considering the challenging nature of SQuAD, our work is based on SQuAD v1.1 benchmark.

In order to perform the evaluation of MC models, two different metrics are utilized: Exact Match (EM) and a softer metric, F1 score (Jurafsky and Martin, 2014). EM measures the percentage of predictions that match any one of the ground truth answers exactly. F1 score measures the average overlap between the prediction and ground truth answer.

2.1.2 Notation

In order to define formally the MC task considered in our work, we follow the SQuAD syntax. A Document (paragraph) and a Question are given as inputs. The Document is a sequence of m words $(w_1^p, w_2^p, ..., w_m^p)$ and a Question is another sequence of n words $(w_1^q, w_2^q, ..., w_n^q)$. The output is a set $\{a^s, a^e\}$, where $1 \leq a^s \leq a^e \leq m$ and a^s, a^e are the boundaries of the answer span, this means that $(w_{a^s}^p, w_{a^s+1}^p, ..., w_{a^e}^p)$ is the answer extracted from the Document sequence. It can be seen in the example extracted from SQuAD (Rajpurkar et al., 2016) shown in Table 2.1.

2.2 Model Pipeline

This section presents a generic architecture that most models have to perform MC (Hu et al., 2018; Wang et al., 2017; Seo et al., 2017) . Generally, the current models are composed by three modules: Context Encoder (CE), Interaction Encoder (IE) and Answer Decoder (AD).

In Figure 2.1 we can see an overview of a generic pipeline. The information flows from bottom to top. First, the model is fed by two inputs—Document and Question—Then, the CE processes the inputs generating a context encoding. The IE merges them, generating an interaction encoding. Finally, the AD extracts an answer to the Question from the Document.

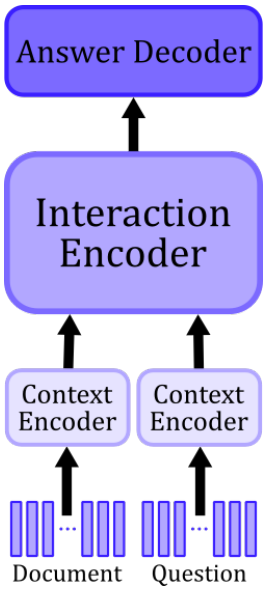


Figure 2.1: MC pipeline

2.3 Input

Frequently, MC models are fed with two inputs, the Document and the Question. The Document is a large word sequence (e.g. a paragraph, a section, an article, etc.) that must be read. Usually, the Document contain knowledge about any topic. However, this knowledge is relevant to answer some questions. On the other hand, the Question is also another word sequence related to the Document. Basically, The Question is the way that user interact with machine.

In a vast majority of Natural Language Processing (NLP) models (Jurafsky and Martin, 2014; Bahdanau et al., 2015; Seo et al., 2017), the first step is to transform each word in a word sequence with its corresponding word embedding. Frequently, DL-based models encodes word sequences as sequence of word embeddings.

2.3.1 Word Embedding

A word embedding, $E : words \rightarrow \mathbb{R}^n$, is a parameterized function that maps words from some language to a high-dimensional vector space, usually between 50 and 300 dimensions (Pennington et al., 2014). Typically, the function is a lookup table, parameterized by a matrix θ, with a row for each word: $E(w_i) = \theta_i$. For example, we might find:

$$E(\text{"}thinks\text{"}) = (0.7, 0.2, -0.4, ...)$$
$$E(\text{"}things\text{"}) = (0.3, -0.8, 0.9, ...)$$

Word embedding exhibit a remarkable property: analogies between words seem to be encoded in the difference vectors between words. For example, there seems to be a constant male-female difference vector:

$$E(\text{"}king\text{"}) - E(\text{"}queen\text{"}) \simeq E(\text{"}man\text{"}) - E(\text{"}woman\text{"})$$
$$E(\text{"}mice\text{"}) - E(\text{"}mouse\text{"}) \simeq E(\text{"}wolves\text{"}) - E(\text{"}wolf\text{"})$$
$$E(\text{"}Peru\text{"}) - E(\text{"}Lima\text{"}) \simeq E(\text{"}France\text{"}) - E(\text{"}Paris\text{"})$$

The word embedding learned to encode gender in a consistent way. In fact, there is probably a gender dimension. Same thing for singular vs plural. It turns out that much more sophisticated relationships are also encoded in this way.

2.4 Context Encoder (CE)

The Context Encoder (CE) is responsible for encoding the words according to their current context. Frequently, some words usually have many meanings determined by the context. The aim of CE is to encode each word given the context, their surrounding words.

Accordingly to the aim, we feed a deep neural network. This can be some kind of Recurrent Neural Network (RNN), a structured set of these, or RNNs combined with another architectures, for instance: stacked RNNs or RNNs combined with Highway layers. Later, we explain RNNs and Highway Networks in detail.

In such a way that the output is the encoding of each word with respect to its current context (Yang et al., 2017; Zhang et al., 2017; Liu et al., 2017). The context encoding of the document and question are generally generated independently, although the same CE is shared for both.

2.4.1 Highway Networks

A Highway Network is a special kind of deep neural architecture, a Highway layer smartly merges the input with a candidate input (Srivastava et al., 2015a). First, we

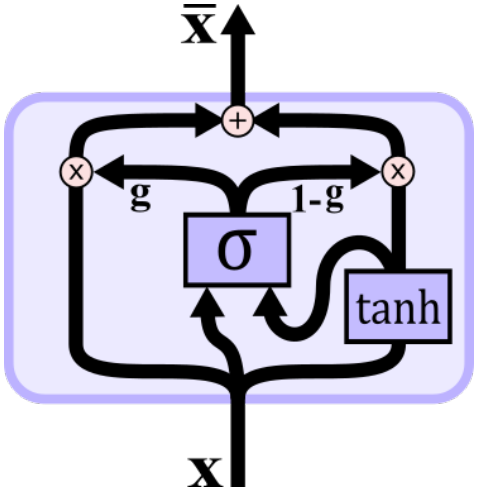


Figure 2.2: Flow diagram of Highway networks.

computed the candidate input $\bar{x}$ according to Equation 2.1, where the bias b and W are trainable parameters that linearly transform the input vector x^{in}. Then, a non-linear function f is applied in order to obtain $\bar{x}$, generally *tanh* is used.

$$\bar{x} = f(W \cdot x^{in} + b) \tag{2.1}$$

Then, we computed a gate to control the flow of information. Essentially, a gate is a real number between 0 and 1, such that multiplying with an input determines how much of this input passes to the next layer. If the gate is close to 1, the input pass completely. On the other hand, when gate is close to 0, the input does not pass to the next layer.

Equation 2.2 defines the Highway gate, this gate is computed based on the input x^{in} and the candidate input $\bar{x}$, which are transformed linearly by their trainable parameters W_x, $W_{\bar{x}}$ and b_g. Finally, the sigmoid (σ) function maps the numbers so that they are between 0 and 1.

$$g = \sigma(W_x \cdot x^{in} + W_{\bar{x}} \cdot \bar{x} + b_g) \tag{2.2}$$

The core idea behind Highway Networks is to control the flow of two information lines with one gate, merging both information flows. Equation 2.3 defines formally this concept.

$$x^{out} = g \times x^{in} + (1 - g) \times \bar{x} \tag{2.3}$$

Where $\times$ is the element-wise product, $(1 - g)$ is the complement of the gate g, forming two complementary gates. The gate g controls the flow of the input x^{in}, while the complementary gate $(1 - g)$ controls the flow of the candidate input $\bar{x}$. Finally, both information flows are element-wise summed in order to merge them, obtaining the output vector x^{out}. Figure 2.2 represents this process.

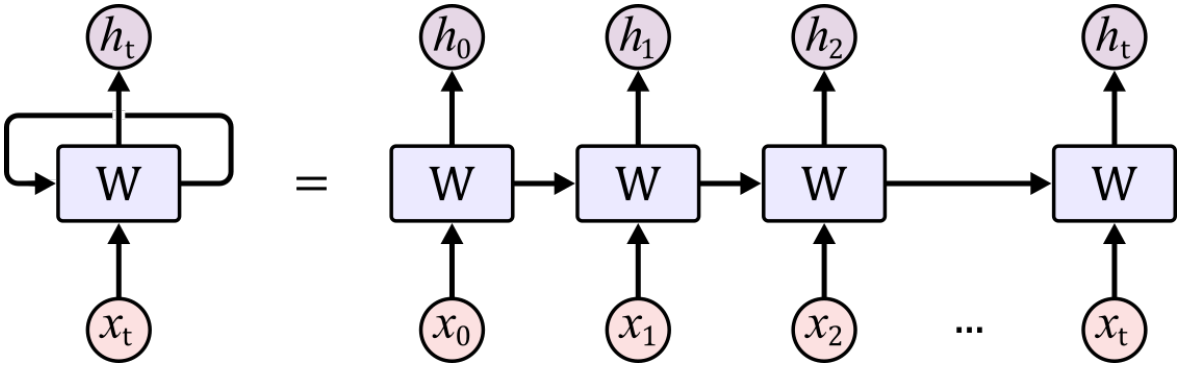


Figure 2.3: RNN can be unrolled and treated as a feed-forward neural network

2.4.2 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence (see figure 2.3). This allows it to exhibit dynamic temporal behavior for a time sequence (Goodfellow et al., 2016, Chapter 10). Unlike feed-forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

The RNNs are frequently represented mathematically (see equation 2.4). Where h_{t-1} represents the previous state, x_t is the input vector at some time step and b is a bias. On the other hand, W_h and W_x are trainable parameters related to h_{t-1} and x_t respectively. In such a way that when being transformed with some function f, such as: $tanh$ or sigmoid (σ), this generates a new state h_t.

$$h_t = f(W_h \cdot h_{t-1} + W_x \cdot x_t + b) \tag{2.4}$$

Given the linearity of equation 2.4 this can be simplified as shown in equation 2.5, where W is a trainable parameter and $[h_{t-1}, x_t]$ is the concatenation of vectors h_{t-1} and x_t.

$$h_t = f(W \cdot [h_{t-1}; x_t] + b) \tag{2.5}$$

RNNs are quite useful when modeling sequences. However, they have some limitations, especially when dealing with long-term dependencies. Often, when training RNNs we could have exploding or vanishing gradient problems. Exploding gradient happens when the gradient grows to infinity in each step and the memory becomes unstable. It is usually solved by limiting the gradient with a threshold. Vanishing gradient is the opposite, it happens when the gradient decreases in time very quickly to 0. It means that the memory was not able to remember in the long term. This problem of long-term dependencies was explored in depth by Hochreiter (1991) and Bengio et al. (1994).

2.4.3 Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) is a special kind of RNN, capable of dealing with long-term dependencies. LSTMs were introduced by Hochreiter and Schmidhuber (1997) in order to deal with the vanishing gradient problem when training traditional RNNs. A common LSTM unit is composed of a cell state and three gates to control and protect the cell state: an input gate, an output gate and a forget gate. In Figure 2.4, we show the inner architecture of an LSTM.

2.4.3.1 Cell State

The main idea behind LSTM is an additional memory called cell state, represented by C_t. This cell state saves the information through time, which can only be updated through several control mechanisms called gates.

2.4.3.2 Forget Gate

The forget gate decides what information should be removed from the cell state. This is basically a sigmoid layer, whose output is a set of numbers between 0 and 1. Intuitively, a 1 means "keep the memory" while a 0 means "clear memory".

$$f_t = \sigma(W_f \cdot [h_{t-1}; x_t] + b_f) \tag{2.6}$$

Where h_{t-1} represents the previous hidden state, x_t is the input vector in step t, W_f is a set of trainable parameters, b_f is the bias and σ is the sigmoid function.

2.4.3.3 Input Gate

The input gate regulates the flow of information to update the cell state from the current input (see equations 2.7). The first equation represents the input gate, where unlike the forget gate, W_i is another set of trainable parameters and b_i is another bias. The second equation generates a new candidate vector in order to update the cell state called $\tilde{C}_t$.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}; x_t] + b_i) \\ \tilde{C}_t &= tanh(W_{\tilde{C}} \cdot [h_{t-1}; x_t] + b_{\tilde{C}}) \end{aligned} \tag{2.7}$$

Now the previous cell state, C_{t-1}, is updated in order to get the new cell state C_t (see equation 2.8). First, the previous cell state is multiplied by f_t, cleaning or saving the memory in this way. Then we add $i_t \times \tilde{C}_t$. In order to save the current candidate input controlled by the input gate.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.8}$$

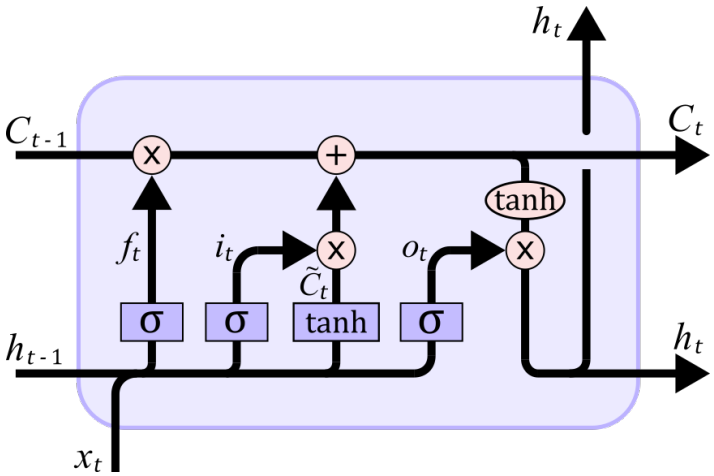


Figure 2.4: LSTM inner structure.

2.4.3.4 Output Gate

Finally, the output gate is responsible for regulating the flow of information from the cell state to the output (see equations 2.9). The first equation represents the gate itself. where similar to previous gates. It has its own set of trainable parameters W_o and its bias b_o. The second equation gives us the new hidden state h_t based on the current cell state C_t and regulated by the output gate o_t.

$$\begin{aligned} o_t &= \sigma(W_o \cdot [h_{t-1}; x_t] + b_o) \\ h_t &= o_t \times tanh(C_t) \end{aligned} \tag{2.9}$$

RNNs are used throughout the model. In most models, RNNs are also used in IE and AD.

2.5 Interaction Encoder (IE)

The Interaction Encoder (IE) merges the context encoding of the document and the question. IE is responsible for obtaining a Question-aware Document representation. Frequently, attention is used to encode the interaction between the Question and the Document (Seo et al., 2017; Xiong et al., 2017; Dhingra et al., 2017). This attention can be given in only one direction, in order to focus the attention in parts of the document according to the question, or in both directions. Recently, self-attention is used as a second step of reasoning (Wang et al., 2017), which is to focus attention on parts of the previous interaction encoding based on itself.

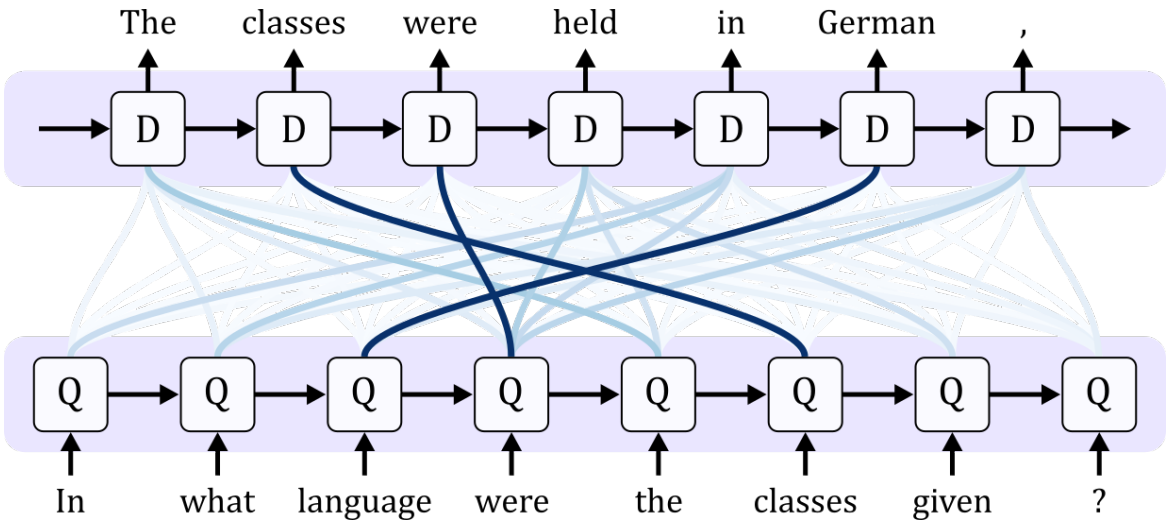


Figure 2.5: Attention encoder applied to Machine Comprehension task. Above we can see the the document encode, while below we have the question encode.

2.5.1 Attention

Psychology tells us that attention is a behavioral and cognitive process of selective concentration in a discrete aspect of information, while other perceptible aspects are ignored (Anderson, 2005). Neural networks can mimic this same behavior, focusing on a subset of the information they are given. For example, a RNN can attend over the output of another RNN. At every time step, it focuses on different positions in the other RNN. The attending RNN generates a query describing what it wants to focus on. Each item is dot-producted with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

This type of attention between RNNs has a number of applications, for example: In Machine Translation (MT), allowing one RNN to process the English input and then have another RNN skim over it, focusing on the word that is currently being translated (Bahdanau et al., 2015). In voice recognition, allowing one RNN to process the audio and then have another RNN skim over it, focusing on relevant parts as it generates a transcript (Chan et al., 2016).

In MC, one RNN processes the question while another RNN focuses on the most relevant words to have an answer (Wang et al., 2017). In figure 2.5 we can see a diagram of how it works, given the question: In what language were the classes given?, we can notice that when encoding the attention distribution for the word *were* this pays more attention to the word *were* of the document. Mainly, This is because they are the same word. However, when the word *language* is processed, the distribution of attention favors the word *German*, which in this case is the answer to the question. The attention distributions were extracted from (Wang and Jiang, 2017).

2.5.2 Bi-Directional Attention Flow (BiDAF)

Bi-Directional Attention Flow (BiDAF) (Seo et al., 2017) is an attention mechanism. BiDAF couples the Document and Question representations, in order to produce a Question-aware representation for each word in the Document.

Let $P = [p_1, p_2, ..., p_m] \in R^{d \times m}$ the contextual representation for each word in Document. On the other hand, $Q = [q_1, q_2, ..., q_n] \in R^{d \times n}$ is contextual representation for n words in Question. The first step is to compute a similarity matrix $S \in R^{m \times n}$, where each s_{ij} indicates the similarity between i-th Document word and j-th Question word. Similarity is formally defined in Equation 2.10.

$$s_{ij} = W_s \cdot [p_i; q_j; p_i \times q_j] \in R \tag{2.10}$$

Where $W_s \in R^{1 \times 3d}$ is a trainable parameter, $\times$ is element-wise product and [;] is concatenation across row. The next step is to compute the attention in both directions using S matrix.

2.5.2.1 Document-to-Question Attention

Document-to-Question (D2Q) attention can be interpreted as: what Question words are most relevant to each word of the Document?. Softmax function is applied to each column of $S^\top$, giving us attention weights (A^{D2Q}) according Document words. Then attention weights are applied to Question context representation (Q), in order to obtain the attended Question representations for all Document words (I^{D2Q}). Equations 2.11 represent D2Q attention.

$$\begin{aligned} A^{D2Q} &= softmax_{col}(S^\top) \in R^{n \times m} \\ I^{D2Q} &= Q \cdot A^{D2Q} \in R^{d \times m} \end{aligned} \tag{2.11}$$

2.5.2.2 Question-to-Document Attention

Question-to Document (Q2D) attention means that the Softmax function is applied to the vector of the most similar word in Question for each Document word, obtaining attention weights (A^{Q2D}). This attention weights are applied to Document context representation (P), in order to obtain a kind of Document summary representation ($\tilde{I}^{Q2D}$). This Document summary is tiled m times for representing Q2D representation (I^{Q2D}). We denote it formally in Equations 2.12.

$$\begin{aligned} A^{Q2D} &= softmax_{row}(max_{col}(S)) \in R^{m \times 1} \\ \tilde{I}^{Q2D} &= P \cdot A^{Q2D} \in R^{d \times 1} \\ I^{Q2D} &= tile_{col}(\tilde{I}^{Q2D}, m) \in R^{d \times m} \end{aligned} \tag{2.12}$$

Finally, the concatenation shown in Equation 2.13 denotes the Question-aware Document representation.

$$I = [P; I^{D2Q}; P \times I^{D2Q}; P \times I^{Q2D}] \in R^{4d \times m} \tag{2.13}$$

Where $\times$ denotes the element-wise product, P is the Document context representation, I^{D2Q} is the D2Q attention and I^{Q2D} is the Q2D attention.

2.6 Answer Decoder (AD)

The Answer Decoder (AD) extracts a piece of text from the document to answer the question. For this, *pointer networks* (Vinyals et al., 2015) are generally used. Later, we explain *pointer networks* in detail. Summing up, they have the ability to learn the conditional probability of a sequence based on another, thus being able to point to a position in the sequence. Usually two pointer networks are used to determine the boundaries of the answer given the interaction encoding (Wang and Jiang, 2017; Xiong et al., 2017; Kadlec et al., 2016).

2.6.1 Pointer Networks

Vinyals et al. (2015) introduced a new neural network architecture called pointer networks. It learns the conditional probability of an output sequence with elements that are discrete tokens corresponding to position in an input sequence. This problem can't be solved by previous architectures, because the number of target classes in each step of the output depends on the variable length of the input. Pointer networks can be represented mathematically as follows:

$$\begin{aligned} u_i &= f(W \cdot [u_{i-1}; x_i] + b) \\ p(C_j|C_1, ..., C_{j-1}) &= softmax(u_j) \end{aligned} \tag{2.14}$$

Where u_i is a RNN hidden state at time step $i \in (1, ..., n)$, n is the length of the sequence. u_i is a $j-$dimensional vector, then $softmax$ normalizes the vector u_j (of length n) to be an output distribution over the n input vectors.

2.7 Summary

In this chapter, we presented the background related to MC. We defined the MC task, their three different annotation styles and metrics. we also presented several deep neural architectures, organized by their influence in each stage of the pipeline. In the next chapter the related work is presented, we organize the related work for their contributions to the pipeline stages.

Chapter 3

Related Work

This chapter presents the works related to DL-based models for performing MC task. This chapter is organized basing on contributions on the pipeline stages. A general scheme of the pipeline would consist of three parts. The first one encodes the Document and the Question into fixed-size vectors. The second one encodes the interaction between the Question and the Document. Finally, the third one is responsible for extracting the answer. Some of these models inspired our work.

3.1 Contributions Related to Inputs

The vast majority of models are fed with word embeddings. Frequently, they used Global Vectors (GloVe) (Pennington et al., 2014), a group of them used 100-dimension embedding that was pre-trained with a 6B corpus (Hu et al., 2018; Gong and Bowman, 2018; Seo et al., 2017) , while a bigger group used 300-dimension GloVe pre-trained with 840B Common Crawl corpus (Yu et al., 2018; Salant and Berant, 2018; Huang et al., 2018; Xiong et al., 2018; Shen et al., 2017b).

A great variety of models included character-level embedding to deal with out-of-vocab words. Many models followed work of Kim (2014), computing character embedding with Convolutional Neural Network (CNN)s (Yu et al., 2018; Salant and Berant, 2018; Gong and Bowman, 2018; Shen et al., 2017b; Seo et al., 2017). On the other hand, some models used RNN-based character embedding (Hu et al., 2018; Wang et al., 2017). These taken the last hidden states of a RNN.

Due to character-level embedding successful, some models added another word features in embedding (e.g. Part-Of-Speech (POS), Named Entity Recognition (NER), Term Frequency (TF)) . This simple pre-trained features added to input embedding, showed a relative improvement over models without them (Hu et al., 2018; Huang et al., 2018; Liu et al., 2018; Shen et al., 2017b).

3.2 Contributions in Context Encoder (CE)

This section presents related work with contributions to CE. Usually, models are based on RNN combined with other neural architectures.

Seo et al. (2017) follow a multi-stage hierarchical process that represents the Document and Question at different levels of granularity. First, word-level and character-level embeddings are concatenated. Then, they applied two-layer Highway Network (Srivastava et al., 2015b) to obtain a new Document and Question representations. Finally, these representations fed an LSTM (Hochreiter and Schmidhuber, 1997) for obtaining their context encodings. Our work had this CE as baseline.

Yang et al. (2017) proposed a fine-grained gating mechanism to dynamically combine word and character embeddings based on properties of the words. They fused word-level and character-level embeddings with a Highway layer, whose gate is based on additional features (POS, NER and frequency). Their results showed improvements in several NLP tasks. Our work proposes a similar approach for fusing Document representations.

Weissenborn et al. (2017) introduced a simple context/type matching heuristic, which is based on the first word of the Question. This architecture is too simple despite having acceptable results.

Chen et al. (2017) proposed a Document Reader, which included three simple binary features based on exact match metric. This simple aggregation showed to improve the performance.

Liu et al. (2017) proposed to use structured linguistic information such as: constituency trees and dependency trees. They used a LSTM to encode a variable-length syntactic sequence into a fixed-length vector representation. We found that this model can perform especially well on exact match metrics, which requires syntactic information to accurately locate boundaries of answers.

Salant and Berant (2018) presented a contextualized word representation inside CE. They used a Highway (Srivastava et al., 2015b) layer to merge contextual and non-contextual representations. On the one hand, word embedding represents non-contextual information. On the other hand, contextual representation is a re-embedding based on character-level embedding and a strong Language Model (Józefowicz et al., 2016), that was pre-trained on the One Billion Words Benchmark (Chelba et al., 2013). This setup improved largely its baseline model.

Huang et al. (2018) defined history-of-word (HoW) as the concatenation of all the representations generated for each word across the model. For context encoding, they proposed to use two bidirectional LSTM layers to generate low-level and high-level concepts. Additionally, Question encoder has another bidirectional LSTM layer in order to get a Question Understanding. It is fed with previous HoW (concatenation of low-level and high-level representations). Their results showed a positive impact of

their HoW approach. Our work is based on a similar approach to HoW.

Yu et al. (2018) proposed a similar architecture to BiDAF (Seo et al., 2017). However, they proposed to replace RNNs with convolutions and self-attention, where convolution models local interactions and self-attention models global interactions. For self-attention, they used the multi-head attention defined by Vaswani et al. (2017). The main advantage of convolution+self-attention with respect to RNNs is its training speedup, due to that convolutions are highly parallel. This also allowed them to train a bigger model.

Recently some works (Hu et al., 2018; Huang et al., 2018; Xiong et al., 2018; Salant and Berant, 2018) apply Transfer Learning including Language Models (Peters et al., 2018; McCann et al., 2017; Józefowicz et al., 2016)

3.3 Contributions in Interaction Encoder (IE)

This section describes several IEs. Frequently IEs are based on Attention. We organize this section based on time and architectures progression.

Along with cloze style datasets, several powerful DL models (Hermann et al., 2015; Hill et al., 2015; Kadlec et al., 2016; Sordoni et al., 2016; Trischler et al., 2016) have been introduced to solve cloze style benchmarks. The Attentive Reader (Hermann et al., 2015) was the first to introduce attention mechanism into reading comprehension style QA. Hill et al. (2015) proposed a window-based memory network for their Children's Book Test (CBT) dataset.

Another key component of DL models is the attention-based RNN, which has demonstrated success in a wide range of tasks. Bahdanau et al. (2015) first propose attention-based recurrent networks to infer word-level alignment when generating the target word. Hermann et al. (2015) introduced word-level attention into reading comprehension to model the interaction between Questions and Documents. Rocktäschel et al. (2016) and Wang and Jiang (2016) proposed determining entailment via word-by-word matching.

Weighted attending to Document words has been proposed in several works. Ling et al. (2015) proposed considering window-based contextual words differently depending on the word and its relative position. Cheng et al. (2016) proposed a novel LSTM network to encode words in a sentence which considers the relation between the current token being processed and its past tokens in the memory. Parikh et al. (2016) apply this method to encode words in a sentence according to word form and its distance. Since Document information relevant to Question is more useful to infer the answer in reading comprehension.

Alternating Iterative Attention (Sordoni et al., 2016) proposed an iterative attention mechanism to better model the links between Question and Document. The

EpiReader (Trischler et al., 2016) solved cloze style QA task by combining an attentive model with a re-ranking model.

Attention-over-Attention Reader (Cui et al., 2017) proposed a two-way attention mechanism to encode the Document and Question mutually, inspiring the next models generation. Gated-Attention Reader (Dhingra et al., 2017) proposed iteratively selecting important parts of the Document by multiplying gating function with the Question representation, with clearly satisfactory results.

Wang and Jiang (2017) proposed an architecture based on match-LSTM (Wang and Jiang, 2016), a model for predicting textual entailment. In MC context, the match-LSTM goes through the words of the Document sequentially. At each position of the Document, attention mechanism is used to obtain a weighted vector representation of the Question. This weighted Question is then to be combined with the current word representation of the Document and fed into an LSTM.

Xiong et al. (2017) proposed a co-attention mechanism that attends simultaneously to the Document and Question. They computed a similarity matrix by dot product between word representations in Document and Question. Then, these scores are multiplied by Document and Question representations. Finally, co-attention representation is passed through LSTM to fuse of temporal information.

Seo et al. (2017) introduced Bi-Directional Attention Flow (BiDAF), an attention mechanism that achieved a Question-aware Document representation without early summarization. First, a function with trainable parameters computed the similarity between words in Document versus Question, generating a similarity matrix. The similarity matrix is used to generate Document-to-Question and Question-to-Document attentions. Then, these are fused by concatenation. In their ablation study, Document-to-Question attention probed to be critical with an accuracy drop of more than 10 points. Our work uses BiDAF as IE.

Pan et al. (2017) proposed MEMEN, a multi-layer embedding to encode the Document and the memory network of full-orientation matching to obtain the interaction of the Document and Question. They used memory hops to refine their answer.

Wang et al. (2017) proposed a gated attention-based RNN to incorporate Question information into Document representation. It is similar to match-LSTM (Wang and Jiang, 2016) with an additional gate to determine the importance of Document words regarding the Question. They also proposed a self-matching attention layer over attention layer, self-attention is an attention mechanism (could be the same) whose inputs are the same (Document representation and Document representation). It dynamically refines the Document representation by looking over the whole Document and aggregating evidence relevant, allowing the model to make full use of Document information. Their ablation tests highlighted the importance of self-attention layer.

Clark and Gardner (2018) added self-attention to BiDAF (Seo et al., 2017) model. For self-attention, they used the same attention mechanism without Question-to-Document attention. Then, attention and self-attention are fused in a residual setup,

self-attention is summed with previous attention encoding. Their results clearly surpass their baseline model.

Gong and Bowman (2018) extended BiDAF IE adding a second attention step. After first attention flow layer, they added a LSTM-based summarization layer that summary all attention in a fixed-size vector. Then, new Document and Question representations are generated based on previous representations and summarized attention. Finally, these new representations are fed into a second attention flow layer.

Xiong et al. (2018) evolved their co-attention encoder. It had two co-attention levels and fused them with concatenation through residual connections. This deep residual co-attention accompanied by a Reinforcement learning (RL) training showed an improvement over their baseline. Additionally, they added Context Vectors (CoVe) (McCann et al., 2017) that was pre-trained with a MT benchmark. This improved largely their previous work.

Huang et al. (2018) proposed a fully-aware attention based on multiplicative attention (Britz et al., 2017). They apply their lightweight attention independently at multiple understanding levels, from embedding to high-level representations. Then, they used the same mechanism to find self-attention of Document representation. In their ablation study showed the great impact of their multi-level attention strategy.

Hu et al. (2018) presented a re-attention mechanism composed by several aligning blocks structured in temporal way. It means each block depends on previous block. Each block computed Document-Question attention and self-attention. For Attention, they used a function that computed a similarity matrix in order to obtain the Question-aware Document Representation. Then, it is fused with Document encoding through highway layer. Similarly, self-attention is computed. Additionally, they proposed a RL optimization method to fine-tune their trained model.

3.4 Contributions in Answer Decoder (AD)

This section explores the different architectures related to AD. The vast majority of works uses Pointer Networks (Vinyals et al., 2015) at the output layer. Given the variable length of sequences, Pointer Networks seem to be the best option to perform the answer decoding compared with traditional neural architectures.

Wang and Jiang (2017) proposed two different approaches to decode answer: The sequence approach used a Pointer Network (Vinyals et al., 2015) to generate a word sequence but these words may not be consecutive in the Document. The boundary approach produced only the start and end word positions, in such a way that all words inside of boundaries could be the answer. Boundary approach surpassed largely the sequence approach, we think that it is due to the benchmark nature. The Benchmark's aim is to extract a continuous span of text as answer. Due to their excellent results, following related works followed the boundary approach.

Xiong et al. (2017) proposed a dynamic pointing decoder which is similar to a state machine whose state is maintained by an LSTM (Hochreiter and Schmidhuber, 1997). The new state is generated based on previous estimate of the start and end positions and previous state. Then, they proposed Highway Maxout Network to compute new scores for determining answer. It is composed by a *tanh* layer and three Maxout (Goodfellow et al., 2013) layers, where first and third are connected by a residual connection. Dynamic decoder iteratively estimates the answer span. Due to its iterative nature, it is able to recover itself from initial local maximum corresponding to incorrect predictions. Their ablation analysis showed the component impact.

Shen et al. (2017a) introduced ReasoNet, a model that improved the AD by proposing a dynamic multi-step reasoning. Powered by RL. Their model is able to determine dynamically when stop the reasoning process. They defined the IE output as memory and Question summary as initial inference state. Then, a termination state is generated based on current inference state. If termination state is false, a new attention is computed based on current inference state and memory. Next, a new inference state is generated by passing current inference state and attention through Gated Recurrent Unit (GRU) (Cho et al., 2014). This process is repeated until termination state will be true. Finally, an answer is generated based on current inference state.

Shen et al. (2017b) performed a deep analysis about multi-step reasoning. Three models were evaluated: single-step, fixed multi-step and dynamic multi-step. Fixed multi-step had 5 reasoning steps, while dynamic multi-step determined dynamically the number of steps. They empirically showed that fixed multi-step reasoning outperforms single-step reasoning by an acceptable margin. However, dynamic multi-step reasoning outperforms fixed multi-step reasoning by narrower margin.

Liu et al. (2018) introduced Stochastic Answer Network (SAN) that simulated multi-step reasoning. This model is strongly focused on AD. First, memory is generated by computing attention and self-attention. In order to find answer boundaries, several predictive distributions are generated based on memory. Then, they applied dropout to the set predictive distributions for discarding some of them. Finally, the boundaries are obtained by averaging the predictive distributions instead of get the best or last. This approach had showed to improve results and their ablation study showed its robustness.

3.5 Summary

In this chapter, we reviewed the state-of-the-art for MC, in terms of benchmarks and deep neural architectures. We made a classification of the related work according to their relevant contributions in the different stages of the pipeline. All the related work found in this chapter inspired us to propose our proposal.

In the following chapter we present our proposal focused on improving the CE. According to the related work, few advances have been made with respect to architectures related to the CE.

Chapter 4

Proposal

This chapter presents the proposal of this thesis based on the related work presented in the previous chapter. We propose a Deep Learning (DL) based model, which focuses on performing the Machine Comprehension (MC) task for the English language. It describes the model following the predominant pipeline in the related work. First, we propose an encoder that seeks the interaction between the Document and the Question in early stages, which consists of two gating mechanism that regulates the flow of information from the Document. Then, we describe the Interaction Encoder (IE) based on attention, which focuses on a subset of the Document representation, where the answer is located. Finally, the Answer Decoder (AD) is responsible for predicting the start and the ending index positions of the answer inside the Document representation. An overview of the model is shown in Figure 4.1.

According to Figure 4.1, the model is fed with two sequence of fixed-size vectors named embeddings. Next, the Context Encoder (CE) denoted in Figure 4.1 as Document Encoder and Question Encoder, CE is responsible for encoding each embedding (word representation) according to its current context, because a word could have several meanings in different contexts. This context is determined by its word neighborhood, thus we use Recurrent Neural Networks (RNNs) to encode the contextual information of each word.

Then, contextual encodings of Document and Question are fused with the IE, in order to obtain a new Document representation, the IE is represented in the middle part of Figure 4.1. We use an attention mechanism whose aim is to highlight words in Document, relevant words to answer the Question. Finally, on the top of the Figure 4.1, we predict the answer to the Question with the AD. We describe formally all the pipeline stages in the following sections.

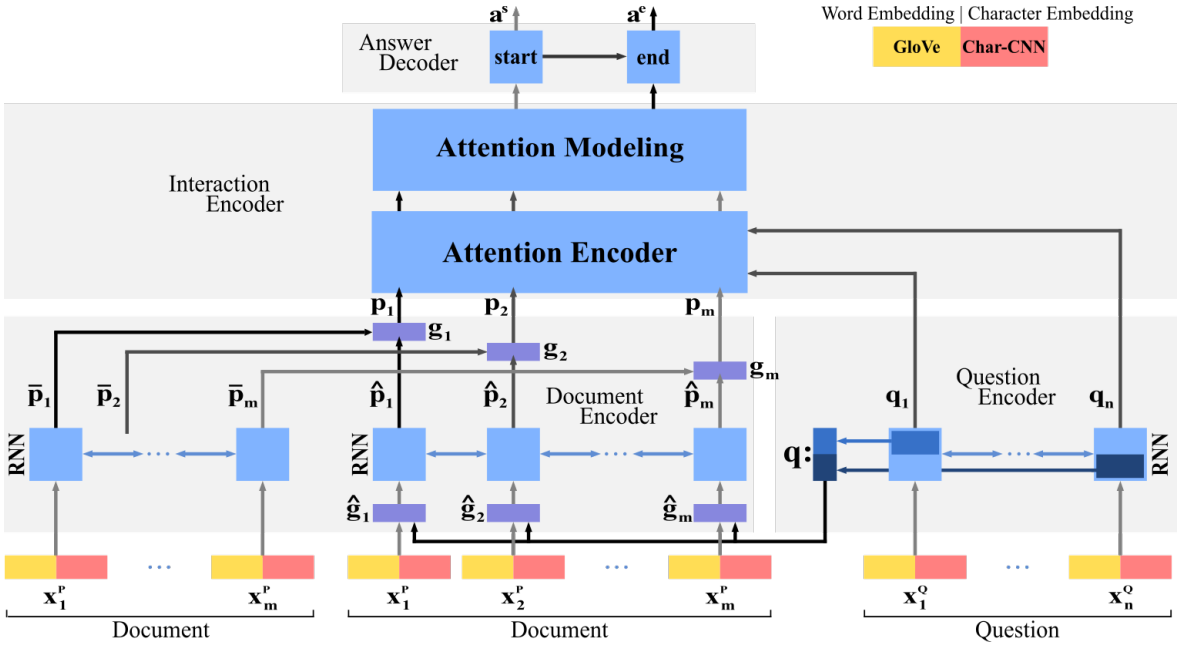


Figure 4.1: Overview of the DL model. The information flows from bottom to top. Below we have the Context Encoder, constituted by: Document Encoder and Question Encoder. In the central part is the Interaction Encoder, and finally up we have the Answer Decoder.

4.1 Input

Our model is fed with two input word sequences, the Document and the Question. The aim in this step is to transform each word of word sequences into word representation, fixed-size vectors. We used two kinds of word representations; Word embedding and character-level embedding.

4.1.1 Word Embedding Layer

The word embedding layer maps each word to a high-dimensional vector space. We use pre-trained word vectors—GloVe (Pennington et al., 2014)—to obtain the fixed word embedding of each word. Let $X^{Qw} = [x_1^{Qw}, x_2^{Qw}, ..., x_n^{Qw}]$ be a vector, which denotes the sequence of n word vectors corresponding to n words in the Question. On the other hand, $X^{Pw} = [x_1^{Pw}, x_2^{Pw}, ..., x_m^{Pw}]$ denote the same for m words in the Document.

4.1.2 Character Embedding Layer

The character-level embedding layer is responsible for mapping each word to a high-dimensional vector space. We computed character-level embedding following the same approach proposed by Seo et al. (2017), which is based on Kim (2014) work. The

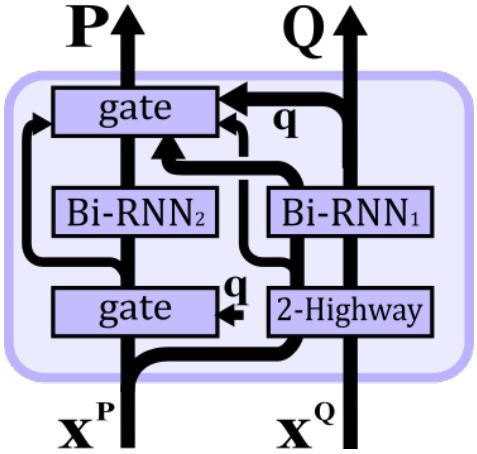


Figure 4.2: Proposed Context Encoder. The information flow in arrow sense, from bottom toward top.

character-level embeddings are generated by a Convolutional Neural Network (CNN). This CNN is applied to embeddings of characters of each word, and then max-pooled in order to obtain a fixed-size vector of each word. Such character-level embeddings had shown to be helpful to deal with out-of-vocab (OOV) words, these words are unknown by the word embedding. Let $X^{Pc} = [x_1^{Pc}, x_2^{Pc}, ..., x_m^{Pc}]$ the character-level embeddings for the m Document words, and $X^{Qc} = [x_1^{Qc}, x_2^{Qc}, ..., x_n^{Qc}]$ represent the sequence of n character-level embeddings for words in the Question.

4.1.3 Embedding Layer

The input Question embedding is obtained by the concatenation of the character-level embeddings and word embeddings, which is denoted by: $X^Q = [x_1^Q, x_2^Q, ..., x_n^Q]$. While, the Document embedding is result of the concatenation of the character-level embeddings and word embeddings, which is formally represented by: $X^P = [x_1^P, x_2^P, ..., x_m^P]$. This input embedding sequences feed their corresponding CEs.

4.2 Context Encoder (CE)

The aim of CEs is to transform the input embeddings—Document and Question—into knowledge represented in sequences of vectors. These vectors can be used by higher layers that extract more abstract knowledge, in order to answer the Question. An overview of our proposed CE is shown in Figure 4.2. Both CEs are in the same block, due to both share some structures, such as: 2-layer Highway network and $Bi\text{-}RNN_1$.

4.2.1 Question Context Encoder

The Question CE is fed with the input Question embeddings denoted by: $X^Q = [x_1^Q, x_2^Q, ..., x_n^Q]$. Then, we applied a 2-layer Highway network expressed by Equation 4.1. In order to obtain a new Question words representation $\bar{X}^Q$.

$$\begin{aligned} \bar{x}_t^Q &= Q_{highway}(x_t^Q) \\ \bar{X}^Q &= [\bar{x}_1^Q, \bar{x}_2^Q, ..., \bar{x}_n^Q] \end{aligned} \tag{4.1}$$

Question encoding is computed by applying a bi-directional RNN to previous representation $\bar{X}^Q$, which is denoted by: $q_t^f = RNN_{forward}^Q(q_{t-1}^f, \bar{x}_t^Q)$ in forward[1], that generates a matrix $Q^f = [q_1^f, q_2^f, ..., q_n^f] \in R^{d \times n}$. Similarly, in backward[2] direction, $q_t^b = RNN_{backward}^Q(q_{t+1}^b, \bar{x}_t^Q)$ generates $Q^b = [q_1^b, q_2^b, ..., q_n^b] \in R^{d \times n}$. Then, we concatenate in order to obtain a Question encoding expressed by Equation 4.2.

$$Q = [q_1, q_2, ..., q_n] \in R^{2d \times n} \tag{4.2}$$

In order to summarize the Question, we concatenate the last hidden states of forward and backward RNNs, which is denoted by Equation 4.3. Where ; denotes concatenation. This Question-summary is an input for our proposed gates in next layers.

$$q = [q_n^f; q_1^b] \in R^{2d \times 1} \tag{4.3}$$

4.2.2 Document Context Encoder

Similarly to the Question CE, the Document CE is fed with the input Document embeddings represented by: $X^P = [x_1^P, x_2^P, ..., x_m^P]$. Next, we applied the 2-layer Highway network to Document embeddings, in order to obtain a high-level representation $\bar{X}^P$, see Equation 4.4.

$$\begin{aligned} \bar{x}_t^P &= D_{highway}(x_t^P) \\ \bar{X}^P &= [\bar{x}_1^P, \bar{x}_2^P, ..., \bar{x}_m^P] \end{aligned} \tag{4.4}$$

First Document encoding is obtained by passing Document embedding through a bidirectional RNN. For simplicity, we will denote the concatenation of RNNs in both directions by Equations 4.5. Where $\bar{P}$ is a sequence of Document contextual encodings, produced by the bidirectional RNN.

$$\begin{aligned} \bar{p}_t &= Bi - RNN_1^P(\bar{p}_{t-1}, \bar{p}_{t+1}, \bar{x}_t^P) \\ \bar{P} &= [\bar{p}_1, \bar{p}_2, ..., \bar{p}_m] \in R^{2d \times m} \end{aligned} \tag{4.5}$$

We propose a second parallel branch of processing, a branch based on gating mechanism accompanied by another Bi-RNN. Our first step was to propose a gating mechanism

[1]It means that steps go from backward to forward.
[2]It means that steps go from forward to backward

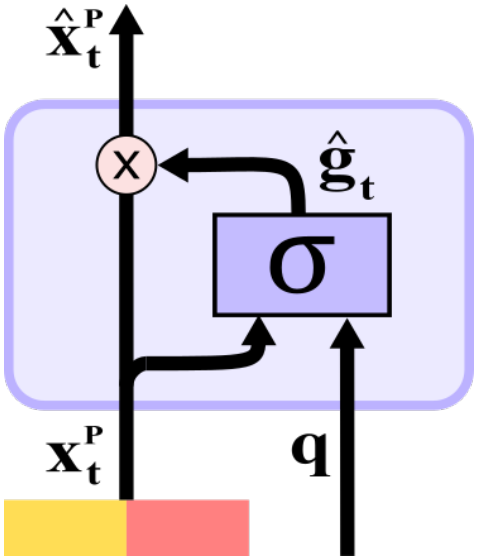


Figure 4.3: Gating mechanism of second processing branch.

represented in Figure 4.3. This gate is based on Question-summary q, with the intention to control the flow of information related to the Question. We define this gating mechanism in Equation 4.6.

$$\begin{aligned} \hat{g}_t &= \sigma(W_{\hat{g}} \cdot [x_t^P; q] + b_{\hat{g}}) \\ \hat{x}_t^P &= \hat{g}_t \times x_t^P \\ \hat{X}^P &= [\hat{x}_1^P, \hat{x}_2^P, ..., \hat{x}_m^P] \end{aligned} \tag{4.6}$$

Where $W_{\hat{g}}$ and $b_{\hat{g}}$ are trainable parameters related to gate $\hat{g}$, q is the Question-summary previously computed in Question CE, and x_t^P is the current Document embedding. The sigmoid function (σ) maps all vector numbers, to the interval between 0 and 1. It allows to regulate the flow: when the gate is closed to 0 information does not pass. On the other hand, when gate is closed to 1, the information pass completely.

We apply the gate to the current input embedding x_t^P, using the element-wise product ($\times$) between gate ($\hat{g}_t$) and current input. It generates a sequence of gated Document embeddings $\hat{X}^P$. Then, we applied a second bidirectional RNN to these gated embeddings. In Equation 4.7, we represent formally this RNN, where $\hat{P}$ is another sequence of Document context encodings.

$$\begin{aligned} \hat{p}_t &= Bi - RNN_2^P(\hat{p}_{t-1}, \hat{p}_{t+1}, \hat{x}_t^P) \\ \hat{P} &= [\hat{p}_1, \hat{p}_2, ..., \hat{p}_m] \in R^{2d \times m} \end{aligned} \tag{4.7}$$

We propose a second gating mechanism for fusing previous context encoding, see Figure 4.4. Gating mechanism is based on Question-summary, previous Document word representations and Document context encodings. In Equation 4.8, we formally define this gating mechanism.

$$g_t = \sigma(W_g \cdot [\bar{p}_t; \hat{p}_t; \bar{x}_t^P; \hat{x}_t^P; q] + b_g) \tag{4.8}$$

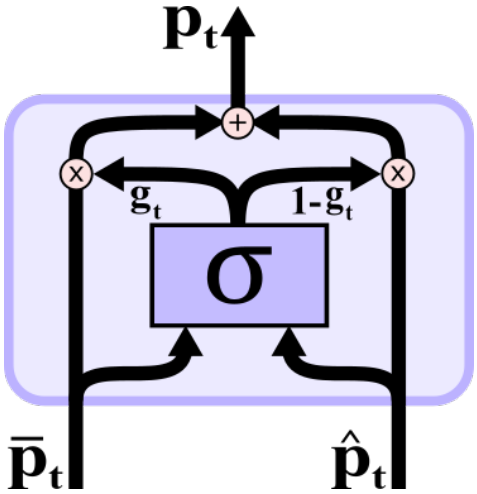


Figure 4.4: Highway-style Gating mechanism for fusing processing branches.

Where W_g and b_g are its trainable parameters. Sigmoid function (σ) computes the gate, based on Document context encoding produced by branches 1 and 2, respectively represented by $\bar{p}_t$ and $\hat{p}_t$, previous Document word representations $\bar{x}_t^P$ and $\hat{x}_t^P$, and Question-summary q computed by Question CE.

Finally, we apply the gate in a Highway-style, similar to the approach proposed by Yang et al. (2017). We control the information flow of two branches, with the same gating mechanism, complementary. Then, we merge by adding operation as can be seen in Equation 4.9.

$$p_t = g_t \times \bar{p}_t + (1 - g_t) \times \hat{p}_t \tag{4.9}$$

Where $\times$ is element-wise product, g_t is the gate and $(1 - g_t)$ is the complementary gate. Thus, the gate controls the flow of branch 1, while complementary gate controls the flow of branch 2. Finally, these are added in order to obtain the final Document context encoding, see Equation 4.10.

$$P = [p_1, p_2, ..., p_m] \in R^{2d \times m} \tag{4.10}$$

Where P is our Document CE output, which is a sequence of vectors, that represents a contextual encoding of each word in Document.

4.3 Interaction Encoder (IE)

In this layer we encode the interaction between the Document and the Question. First, we use Bi-Directional Attention Flow (BiDAF), the attention mechanism proposed by Seo et al. (2017) and widely explained in the Section 2.5.2. It is represented by a function, that fuses the contextual encodings of Document P and Question Q. In order to encode the attention $I \in R^{8d \times m}$.

$$I = Attention(P, Q) \tag{4.11}$$

In order to model the interaction encoding, we pass the attention encoding through two layers of Bi-RNN. These layers capture the interaction between Question-aware Document words representations. Formally, we define this attention modeling in Equation 4.12

$$\begin{aligned} \bar{u}_t &= Bi - RNN_1^I(\bar{u}_{t-1}, \bar{u}_{t+1}, i_t) \\ u_t &= Bi - RNN_2^I(u_{t-1}, u_{t+1}, \bar{u}_t) \\ U &= [u_1, u_2, ..., u_m] \in R^{2d \times m} \end{aligned} \tag{4.12}$$

Where i_t is the current Question-aware Document word representation, computed by attention mechanism. U is another sequence of Document word representations. At this step, Document is completely fused with Question. It will allow us to answer the Question in the next layer.

4.4 Answer Decoder (AD)

We use the AD of BiDAF (Seo et al., 2017). BiDAF AD performs the prediction of starting and ending positions of the answer span (Wang and Jiang, 2017). For this aim, it utilizes two pointer networks (Vinyals et al., 2015), over last Document representations. In Equation 4.13 we compute starting boundary of answer.

$$a^s = softmax(W_{a^s} \cdot [I; U]) \tag{4.13}$$

Where $W_{a^s} \in R^{1 \times 10d}$ is a trainable parameter, I is the attention encoding and U is the output of the attention modeling layer, both were previously computed by IE. Then, our answer starting position will be deduced from a^s distribution. For ending position, we generate another Document representation with another Bi-RNN, see Equation 4.14.

$$\begin{aligned} v_t &= Bi - RNN_1^A(v_{t-1}, v_{t+1}, u_t) \\ V &= [v_1, v_2, ..., v_m] \in R^{2d \times m} \end{aligned} \tag{4.14}$$

Then, we compute the ending boundary following Equation 4.15. Where $W_{a^e} \in R^{1 \times 10d}$ is another trainable parameter, I is the attention encoding and V is the last Document representation. The ending answer boundary is extracted from a^e distribution.

$$a^e = softmax(W_{a^e} \cdot [I; V]) \tag{4.15}$$

Finally, we can predict the answer by extracting from the Document, the text bounded by pointers a^s and a^e.

4.5 Summary

In this chapter, we presented our proposal—a deep neural architecture—focused on improving the CE architecture. We formally described in detail each pipeline stage. In such a way that it is easily replicable.

Our aim proposing this model is to improve a baseline. We will confirm our hypothesis by experiments, and ablations of the proposed model. In next chapter, we will present the results of our proposal.

Chapter 5

Experiments and Results

This chapter presents our experiments and results. In first section we show the setup of the experiments. Next sections describes experiment sets, focusing on improving the Context Encoder (CE). Each set of experiments has its associated results. Finally, In summary section, we present an exhaustive analysis of the experiments and its better single results.

5.1 Setup

In this section, we describe the overall setup of our experiments. The benchmark is presented. A brief description of our baseline model focused on CEs is given. Common implementation details for all our experiment sets are given.

5.1.1 Benchmark

We evaluate the different models on version 1.1 of the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). SQuAD v1.1 contains about 100k+ crowdsourced question/answer pairs, related to 23215 paragraphs extracted from 536 Wikipedia articles. This dataset was partitioned randomly into a training set (80%), a development set (10%), and a test set (10%). SQuAD v1.1 uses two metrics for the evaluation: Exact Match and F1 score. The aim in SQuAD v1.1 is to answer a Question by extracting a span of text from Document (paragraph).

We can find some brief statistics about the benchmark in Table 5.1, a distribution according to the answer type. In addition, we did a length analysis on the training set. The average number of tokens in Document sequences is 138. The Questions have an average of 11.5 tokens. Meanwhile, the answers usually have an average of 3.5 tokens.

Answer type	Percentage	Example
Date	8.9%	19 October 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quietly

Table 5.1: Diversity of answer in SQuAD v1.1 (Rajpurkar et al., 2016).

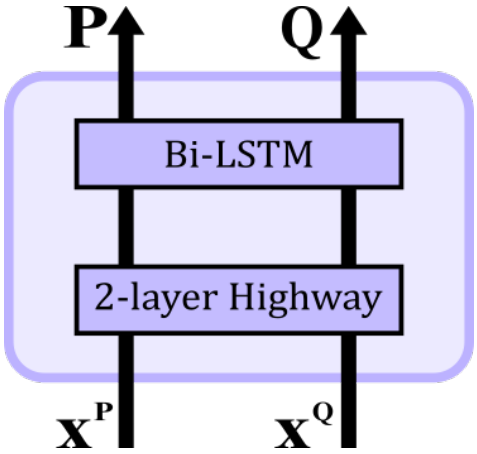


Figure 5.1: Baseline Context Encoder.

5.1.2 Baseline

Our work is based on Bi-Directional Attention Flow (BiDAF) (Seo et al., 2017). We chose BiDAF as a baseline model, because it is a simple model to understand with good results. In addition, its training time is relatively short compared to more sophisticated models. These characteristics were very favorable since our main aim was to improve the CE.

Seo et al. (2017) defined their model architecture in 6 layers based on hierarchies, we grouped some layers to adapt them to a common pipeline to most models. Baseline model and a vast majority of others have two CEs, Document and Question CEs. BiDAF CEs have the same architecture, these are composed by 2-layer Highway (Srivastava et al., 2015b) network and Bi-Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), see Figure 5.1.

In Figure 5.1, Document and Question CEs are represented in the same block, because both share weights, essentially were the same. But, we consider as Document CE all the structures crossed by the information that flows from X^P to P. Similarly,

for Question CE flows from X^Q to Q.

The published works generally exhibit results of their single model and of a set of them working together (ensemble). We only show the results of the single model, because these results are enough to prove our hypothesis. In this way, we compare our single model just with other single models.

5.1.3 Implementation Details

First, we pre-process the SQuAD v1.1, we tokenize each document and question word sequences using PTB Tokenizer (Manning et al., 2014). Then, we set all common hyper-parameters as follows: all Recurrent Neural Network (RNN)s are LSTMs whose hidden state size is 100. We use Adam (Kingma and Ba, 2015) optimizer with default parameters, initial learning rate of 0.001 and exponential decay rate of 0.999, for 12 epochs, with batch size of 60. We set the dropout (Srivastava et al., 2014) rate to 0.2 for all LSTM layers. Finally, the training process took between 12 and 24 hours in a single Tesla K80 GPU in the Manati cluster[1].

All our results showed in Tables (5.2, 5.3, 5.4, 5.5, 5.6 and 5.7) were obtained by averaging the accuracy of n identical models, with different random initial weights. Also, we compute the standard error based on sample standard deviation, with a 95% confidence interval. In Section 5.6, we perform an statistical significance testing for validating our improvements statistically.

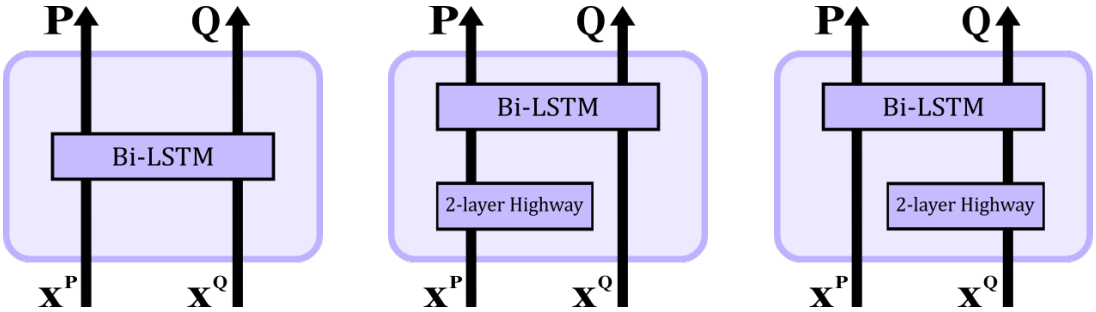
5.2 Small Baseline Ablations

In this section, we show some small variations in the CE of the baseline model, these variations were not documented in the ablation study of the baseline model (Seo et al., 2017). Figure 5.2a shows our first variation, we removed the 2-layer Highway network included in the baseline, we took it out from both CEs. In a second ablation experiment, we removed the 2-layer Highway network just from the Question CE, see Figure 5.2b. Then, our third ablation experiment on baseline, was to remove the 2-layer Highway network from Document CE, see Figure 5.2c.

Table 5.2 shows the results of this experiment set, where *fig* column indicates the graphic representation corresponding to each model, Q_h means that Question CE has the 2-layer Highway network. Similarly, for Document CE D_h. F1 score and Exact Match columns shows mean results with their standard error related to the mean.

The results showed in Table 5.2 were obtained by training three copies for each model. We noticed that third model gave us surprising results, despite the asymmetry.

[1]Manati is a cluster located in Center for High Computational Performance of the Peruvian Amazon. http://iiap.org.pe/web/carcap.aspx



(a) −Both highway (b) −Question highway (c) −Document highway

Figure 5.2: Small ablations in baseline Context Encoder.

Model (Single)	fig	Q_h	D_h	**F1**	**EM**
BiDAF (Seo et al., 2017)	5.1	✓	✓	77.30	67.70
−Both highway	5.2a			77.28 ± 0.16	67.46 ± 0.32
−Question highway	5.2b		✓	77.26 ± 0.21	67.84 ± 0.3
−Document highway	5.2c	✓		**77.68** ± 0.16	**68.13** ± 0.13

Table 5.2: Results of small ablations in baseline Context Encoder.

Given that Document and Question representations are in different vector spaces, due to the 2-layer Highway network just presented in Question CE. The shared Bi-LSTM must be learn to process two different representations with the same weights. Perhaps, the 2-layer Highway network in Question Encoder compensates for the difference in lengths between Document and Question.

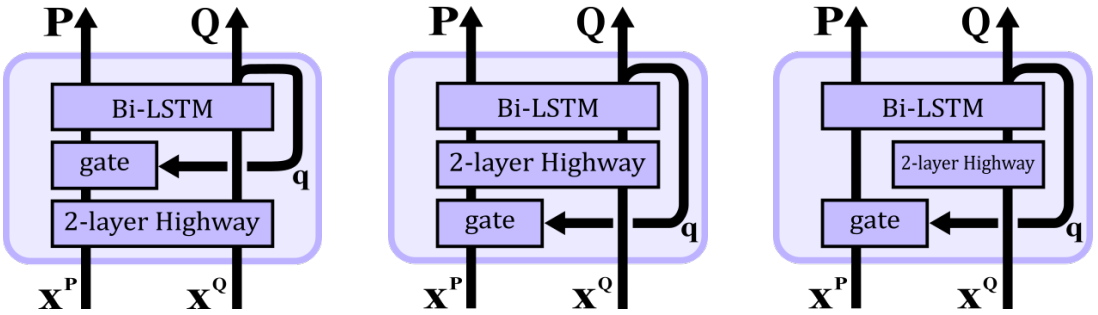
5.3 Gated Baseline

In this set of experiments, we added a particular gating mechanism into the Document CE, this gate is based on question-summary. Formally, this gate is represented by Equation 5.1.

$$\begin{aligned} g_t &= \sigma(W_g \cdot [x_t^{in}; q] + b_g) \\ x_t^{out} &= g_t \times x_t^{in} \end{aligned} \tag{5.1}$$

Where W_g and b_g are trainable parameters, $\times$ is the element-wise product and ; denotes the column concatenation. First, we calculate the linear transformation of input (x_t^{in}) and question-summary (q) with trainable parameters. Then, we compute the gate (g_t) by applying the non-linear sigmoid (σ) function to the linear transformation. Finally, we use the element-wise product between input (x_t^{in}) and gate (g_t), in order to obtain a gated representation (x_t^{out}).

Our first set of experiments were to add this gating mechanism before Bi-LSTM in Document CE. We performed three experiments changing the gate position, we put the gating mechanism after the 2-layer Highway network in Document CE (D_h), see



(a) +Gate after D_h (b) +Gate before D_h (c) +Gate instead D_h

Figure 5.3: Gating mechanism added in three different positions related to 2-layer Highway network of Document Context Encoder (D_h).

Model (Single)	fig	F1	EM
BiDAF (Seo et al., 2017)	5.1	77.30	67.70
+Gate after D_h	5.3a	77.33 ± 0.2	67.85 ± 0.17
+Gate before D_h	5.3b	**77.57** ± 0.09	**67.93** ± 0.18
+Gate instead D_h	5.3c	77.52 ± 0.16	67.93 ± 0.24

Table 5.3: Results of models with gating mechanism in different positions, related to 2-layer Highway network of Document Context Encoder.

Figure 5.3a. Next, we added the gate before D_h, see Figure 5.3b. Finally, we replaced D_h with the gate, see Figure 5.3c.

The results showed in Table 5.3 do not improve our previous best result. However, the third model initially showed a great potential, for this reason we trained this model 10 times. For the other models, we just trained three copies.

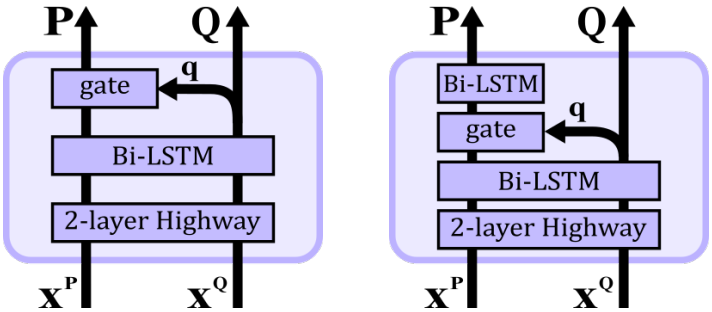
In next experiments we added the gating mechanism after Bi-LSTM of Document CE (D_{LSTM}). We tested two different setups, in the first one, we simply added the gate after D_{LSTM}, see Figure 5.4a. In the second one, we added another Bi-LSTM after the gating mechanism, see Figure 5.4b. Formally, we can complement Equations 5.1 with Equation 5.2.

$$p_t = Bi - LSTM(p_{t-1}, p_{t+1}, x_t^{out}) \tag{5.2}$$

For results in Table 5.4, we trained three copies of each model. We noticed that models with additional Bi-LSTM performed worst. We think that stack another Bi-LSTM adds depth to the model, whereby it needs more training time or another hyper-parameters setup. On the other hand, a scalar gate can be formally restated by Equations 5.3.

$$\begin{aligned} g_t &= \sigma(W_g \cdot [x_t^{in}; q] + b_g) \in R \\ x_t^{out} &= g_t x_t^{in} \end{aligned} \tag{5.3}$$

Where $g_t \in R$ is a scalar. Then, the output (x_t^{out}) is computed by the product of a scalar by an input vector (x_t^{in}).



(a) (b)

Figure 5.4: Gating mechanism added after Bi-LSTM in Document Context Encoder (D_{LSTM}). (a) +Gate or +Scalar gate after D_{LSTM}, (b) +Gate+LSTM or +Scalar gate+LSTM after D_{LSTM}.

Model (Single)	fig	**F1**	**EM**
BiDAF (Seo et al., 2017)	5.1	77.30	67.70
+Gate after D_{LSTM}	5.4a	77.56 ± 0.13	68.06 ± 0.38
+Scalar gate after D_{LSTM}	5.4a	**77.6** ± 0.13	**68.1** ± 0.07
+Gate+LSTM after D_{LSTM}	5.4b	75.71 ± 1.01	66.36 ± 0.95
+Scalar gate+LSTM after D_{LSTM}	5.4b	76.33 ± 0.4	66.92 ± 0.45

Table 5.4: Results of gating mechanism added after Bi-LSTM in Document Context Encoder (D_{LSTM}).

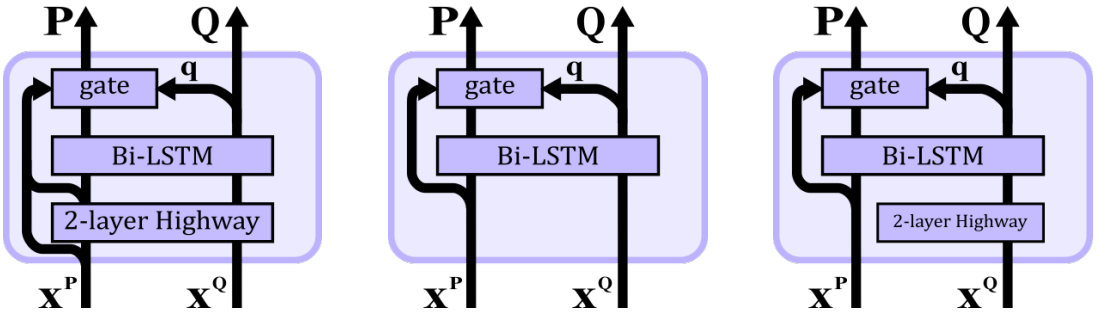
Last experiments on this set also included the gating mechanism after Bi-LSTM of Document CE (D_{LSTM}), see Figure 5.5. However, we redefined the gating mechanism adding new features. Besides to include question-summary, we added all previous representations of document words. Huang et al. (2018) defined the use of previous word representations as history-of-word (HoW), we will use this definition to simplify the concept. Equations 5.4 represent this gate based on question-summary and HoW.

$$\begin{aligned} g_t &= \sigma(W_g \cdot [x_t^{in}; x_t^{HoW}; q] + b_g) \\ x_t^{out} &= g_t \times x_t^{in} \end{aligned} \tag{5.4}$$

Where x_t^{HoW} represents the concatenation of all previous representations of word x_t^{in}, q is the question-summary and x_t^{in} is the current input word representation. W_g and b_g are trainable parameters. then, the gated output (x_t^{out}) is computed by element-wise product ($\times$).

The applied of this gate is showed in Figure 5.5. Inspired by results in first set of experiments, we ablate the model taking out the 2-layer Highway network from Document and Question CEs. When removing it from Document CE, we have only one previous word representation.

Table 5.5 confirms that ablating the 2-layer Highway network just from Document CE improves the results. We noticed that including HoW into the gating mechanism improved the results, comparing with models without it, presented in Table 5.4.



(a) +Gate+HoW (b) +Gate+HoW$-Q_h-D_h$ (c) +Gate+HoW$-D_h$

Figure 5.5: Gating mechanism based on question-summary and history-of-word, added after Bi-LSTM in Document Context Encoder (D_{LSTM}).

Model (Single)	fig	Q_h	D_h	**F1**	**EM**
BiDAF (Seo et al., 2017)	5.1	✓	✓	77.30	67.70
+Gate+HoW after D_{LSTM}	5.5a	✓	✓	77.5 ± 0.15	67.97 ± 0.11
+Gate+HoW after D_{LSTM}	5.5b			77.5 ± 0.22	67.99 ± 0.36
+Scalar gate+HoW after D_{LSTM}	5.5c	✓		77.64 ± 0.19	68.04 ± 0.2
+Gate+HoW after D_{LSTM}	5.5c	✓		**77.73** ± 0.34	**68.16** ± 0.43

Table 5.5: Results of gating mechanism based on question-summary and history-of-word, added after Bi-LSTM in Document Context Encoder (D_{LSTM}).

5.4 Gate+LSTM

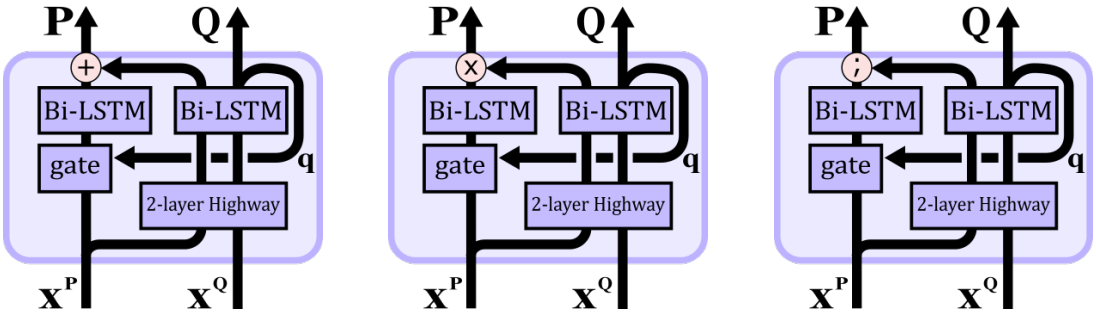
This section presents a set of experiments based on gating mechanism with an additional bi-LSTM. This generates two parallel Document representations, which are then merged by a fusing operation. We represent this architecture formally in Equations 5.5.

$$\begin{aligned} g_t &= \sigma(W_g \cdot [x_t^P; q] + b_g) \\ x_t^{out} &= g_t \times x_t^P \\ \hat{p}_t &= Bi - LSTM_2(\hat{p}_{t-1}, \hat{p}_{t+1}, x_t^{out}) \end{aligned} \tag{5.5}$$

Where x_t^P is the current Document embedding and q is the question-summary. g_t is computed similarly to previous experiments, then we apply the gate to the input embedding. Finally, we use a Bi-LSTM in order to encode another Document contextual encoding. Let $\bar{P} = [\bar{p}_1 ... \bar{p}_m]$ the Document encoding produced by baseline Bi-LSTM, we fused these two Document encodings with an operation represented in Equation 5.6.

$$p_t = \bar{p}_t \circ \hat{p}_t \tag{5.6}$$

Where $\circ$ is the fusing operation. Figure 5.6 presents the model with three different fusing operations. In Figure 5.6a, both Document encodings were fused by element-wise addition (+). In Figure 5.6b, Document encodings were fused by element-wise product ($\times$). In Figure 5.6c, we merged both Document encodings by concatenation (;).



(a) +Gate+LSTM (+) (b) +Gate+LSTM (×) (c) +Gate+LSTM (;)

Figure 5.6: Additional processing branch composed by a gate and Bi-LSTM, with different fusing operations

Model (Single)	fig	F1	EM
BiDAF (Seo et al., 2017)	5.1	77.30	67.70
+Gate+LSTM (+)	5.6a	77.61 ± 0.17	68.13 ± 0.18
+Gate+LSTM (×)	5.6b	**77.76** ± 0.14	**68.34** ± 0.18
+Gate+LSTM (;)	5.6c	77.68 ± 0.2	68.21 ± 0.32

Table 5.6: Results of additional processing branch composed by a gate and Bi-LSTM, with different fusing operations.

Results showed in Table 5.6 demonstrated the effectiveness of this second branch in the Document CE. Our better results in this step were obtained by fusing with element-wise product. However, fusing by concatenation increased the Interaction Encoder (IE) size, also increasing the computational cost.

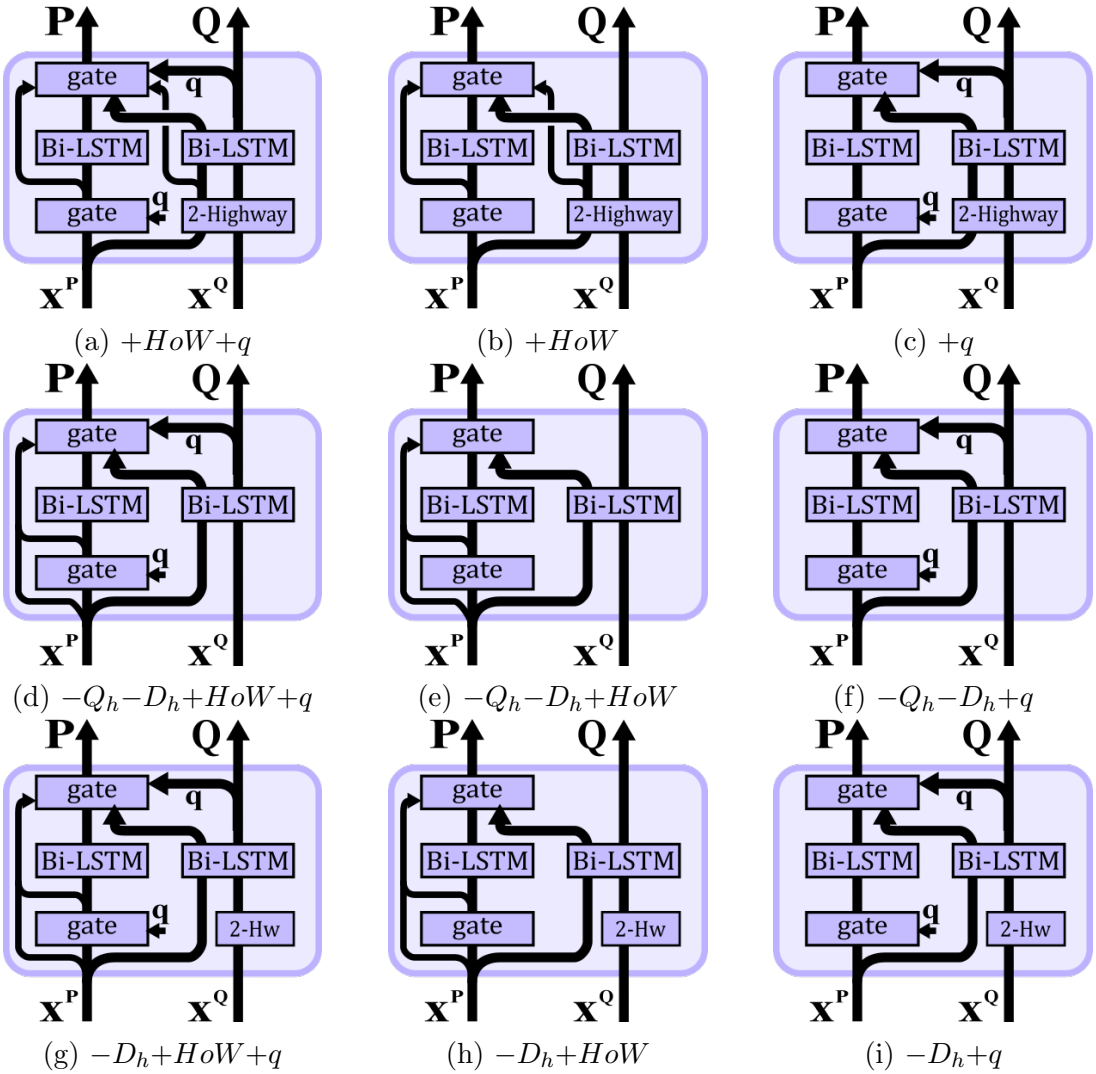
5.5 Two Gated Baseline

This set of experiments expands the previous Gate+LSTM section of experiments. We replace the fusing operation (∘) with another gating mechanism, that merge both Document encodings in a more structured way. We represent this gate in Equations 5.7.

$$\begin{aligned} \tilde{g}_t &= \sigma(W_{\tilde{g}} \cdot [\bar{p}_t; \hat{p}_t; x_t^{HoW}; q] + b_{\tilde{g}}) \\ p_t &= \tilde{g}_t \times \bar{p}_t + (1 - \tilde{g}_t) \times \hat{p}_t \end{aligned} \tag{5.7}$$

Where $\tilde{g}_t$ is a second gate based on both previous Document encodings $\bar{p}_t$ and $\hat{p}_t$, all previous Document word representations x_t^{HoW} and question-summary q. Then, we applied the gate in a Highway fashion to both Document encodings. This gate merge both Document encodings in a smart way, producing a better Document contextual encoding.

Figure 5.7 shows several ablations on our proposed model. In figures 5.7b, 5.7e and 5.7h, we ablate the question-summary q from both gates. In figures 5.7c, 5.7f and



(a) $+HoW+q$ (b) $+HoW$ (c) $+q$

(d) $-Q_h-D_h+HoW+q$ (e) $-Q_h-D_h+HoW$ (f) $-Q_h-D_h+q$

(g) $-D_h+HoW+q$ (h) $-D_h+HoW$ (i) $-D_h+q$

Figure 5.7: Fusing two processing branches with Highway-style gate.

Model (Single)	fig	Q_h	D_h	HoW	q	**F1**	**EM**
BiDAF (Seo et al., 2017)	5.1	✓	✓			77.30	67.70
+Gate+LSTM+Gate	5.7a	✓	✓	✓	✓	77.64 ± 0.23	68.28 ± 0.45
+Gate+LSTM+Gate	5.7b	✓	✓	✓		77.76 ± 0.07	68.33 ± 0.11
+Gate+LSTM+Gate	5.7c	✓	✓		✓	77.62 ± 0.31	68.2 ± 0.25
+Gate+LSTM+Gate	5.7d			✓	✓	77.85 ± 0.22	68.37 ± 0.33
+Gate+LSTM+Gate	5.7e			✓		77.89 ± 0.05	68.49 ± 0.17
+Gate+LSTM+Gate	5.7f				✓	77.65 ± 0.11	68.09 ± 0.12
+Gate+LSTM+Gate	5.7g	✓		✓	✓	77.88 ± 0.23	68.48 ± 0.26
+Gate+LSTM+Gate	5.7h	✓		✓		**78.05** ± 0.1	**68.64** ± 0.16
+Gate+LSTM+Gate	5.7i	✓			✓	77.86 ± 0.31	68.24 ± 0.61

Table 5.7: Results of fusing two processing branches with Highway-style gate.

5.7i, we ablate the previous word representations HoW from the second gate. Figures 5.7d, 5.7e and 5.7f show the ablation of the 2-layer Highway network from both CE. In figures 5.7g, 5.7h and 5.7i, we removed the 2-layer Highway network just from Document CE.

Table 5.7 presents the results of fusing these two processing branches with a Highway-style gate, based on question-summary and previous word representations. Where Q_h is the presence of 2-layer Highway network in Question CE. Similarly, D_h for Document CE. HoW is the use of previous word representations, and q is when gates were based on question-summary.

According to Table 5.7, our best results were obtained when 2-layer Highway network in Document CE were ablated. For this reason we trained 10 copies of these models. For the rest, we trained 3 copies of each model. We noticed that removing question-summary gives stability to the model, given that better average results were obtained. However, the better individual performance was reached by a model with gates based on question-summary.

5.6 Statistical Significance Testing

We perform a Student's t-test with an $\alpha = 0.05$ significance value. We use Independent two-sample t-test, with equal sample sizes of 10 and unequal variances. We define a null hypothesis H_0: there is **NO** significant difference between the mean from the results of different models. On the other hand, our alternative hypothesis is H_1: there is a significant difference between the mean from the results of different models.

Tables 5.8 and 5.9 presents the results of our best mean models. Where t_{score} is the result of t-test between each model and our baseline implementation. Q_h is the presence of 2-layer Highway network in Question CE. Similarly, D_h for Document CE. HoW is the use of previous word representations, and q is when gates were based on Question-summary. In g_2 column, models had a second branch of processing, based on gating mechanism and additional Bi-LSTM. In Column g_3, models use a second Highway-style gate, for fusing both Document context encodings.

Then, we obtained a t_{score} from t-test, we follow the decision criterion: if $t_{score} \leq \alpha$ refuse H_0, else accept H_0. In Table 5.8, we show the t_{score} over F1 measure of our most outstanding models vs baseline. All have a $t_{score} < 0.05$. Thus, we reject the null hypothesis H_0. There is a significant difference between the mean from the results of models vs baseline on F1 score.

Similarly, in Table 5.9 all models have a $t_{score} < 0.05$ rejecting H_0. There is a significant difference between the mean from the results of models vs baseline on EM metric.

Model (Single)	fig	D_h	g_2	g_3	HoW	q	**F1**	t_{score}
Our baseline implementation	5.1	✓					77.54	
+Gate+LSTM (×)	5.6b	✓	✓			✓	77.76	1.4×10^{-2}
+Gate+LSTM+Gate	5.7g		✓	✓	✓	✓	77.88	1.7×10^{-2}
+Gate+LSTM+Gate	5.7h		✓	✓	✓		78.05	5.7×10^{-7}

Table 5.8: Statistical significance testing of our best mean models on F1 score.

Model (Single)	fig	D_h	g_2	g_3	HoW	q	**EM**	t_{score}
Our baseline implementation	5.1	✓					67.95	
+Gate+LSTM (×)	5.6b	✓	✓			✓	68.34	2.3×10^{-3}
+Gate+LSTM+Gate	5.7g		✓	✓	✓	✓	68.48	3.1×10^{-3}
+Gate+LSTM+Gate	5.7h		✓	✓	✓		68.64	3.7×10^{-6}

Table 5.9: Statistical significance testing of our best mean models on Exact Match metric.

5.7 Summary

Experiments showed the effectiveness of the gates influenced by external features, such as: Question-summary and previous Document representations. Also showed the impact of the 2-layer highway network in both CE.

We divided our experiments on four sets. In first set we ablated the baseline CE. In second set we added a gating mechanism into Document CE. In third set we added a second branch into Document CE, and fused these two branches with a simple operation. Finally, in fourth set we fused these two branches with a gating mechanism inspired on Highway networks.

Table 5.10 presents a summary of best single results of all our experiments. Where Q_h is the presence of 2-layer Highway network in Question CE. Similarly, D_h for Document CE. HoW is the use of previous word representations, and q represents gates that were based on question-summary. Column g_1 represents models that included a gate on a single processing branch. In g_2 column, models had a second branch of processing, based on gate and additional Bi-LSTM. In column g_3, models use a second Highway-style gating mechanism, for fusing both previous Document context encodings.

The first set of experiments ablated the 2-layer Highway network from CEs. The best result were obtained when 2-layer Highway network was removed just from Document CE. Apparently this effect is due to the amount of data, given that Document sequence is much longer than Question sequence. In table 5.10, we noticed that models without 2-layer Highway network in Document CE, performed best along all experiment sets. Although, contrasting with mean results exposed in previous tables, we can infer that these models present a great variability.

Model (Single)	fig	Q_h	D_h	g_1	g_2	g_3	HoW	q	**F1**	**EM**
BiDAF (Seo et al., 2017)	5.1	✓	✓						77.30	67.70
Our baseline implement.	5.1	✓	✓						77.61	68.14
−Both highway	5.2a								77.43	67.68
−Question highway	5.2b		✓						77.45	68.12
−Document highway	5.2c	✓							77.81	68.26
+Gate after D_h	5.3a	✓	✓	✓				✓	77.43	67.96
+Gate before D_h	5.3b	✓	✓	✓				✓	77.63	68.03
+Gate instead D_h	5.3c	✓		✓				✓	78.14	68.73
+Gate after D_{LSTM}	5.4a	✓	✓	✓				✓	77.68	68.34
+Gate after D_{LSTM}	5.5a	✓	✓	✓			✓	✓	77.65	68.06
+Gate after D_{LSTM}	5.5b			✓			✓	✓	77.63	68.32
+Gate after D_{LSTM}	5.5c	✓		✓			✓	✓	78.07	68.59
+Gate+LSTM (+)	5.6a	✓	✓		✓			✓	77.76	68.31
+Gate+LSTM (×)	5.6b	✓	✓		✓			✓	78.25	68.70
+Gate+LSTM (;)	5.6c	✓	✓		✓			✓	77.84	68.46
+Gate+LSTM+Gate	5.7a	✓	✓		✓	✓	✓	✓	77.84	68.63
+Gate+LSTM+Gate	5.7b	✓	✓		✓	✓	✓		77.81	68.40
+Gate+LSTM+Gate	5.7c	✓	✓		✓	✓		✓	77.79	68.38
+Gate+LSTM+Gate	5.7d				✓	✓	✓	✓	78.07	68.59
+Gate+LSTM+Gate	5.7e				✓	✓	✓		77.94	68.66
+Gate+LSTM+Gate	5.7f				✓	✓		✓	77.73	68.18
+Gate+LSTM+Gate	5.7g	✓			✓	✓	✓	✓	**78.36**	**69.10**
+Gate+LSTM+Gate	5.7h	✓			✓	✓	✓		78.31	69.02
+Gate+LSTM+Gate	5.7i	✓			✓	✓		✓	78.10	68.72

Table 5.10: Summary of best single results of all our experiments.

On second set of experiment, we added a gating mechanism based on question-summary and previous word representations. This gate was added in different positions into Document CE pipeline. Our best results were obtained putting the gate mechanism after Document bi-LSTM. Table 5.10 shows that best single model in this set, were when we replace the 2-layer Highway network with a gating mechanism. However, this model had showed to be very unstable.

Third experiment added a new processing branch into the Document CE. This branch is composed by a gating mechanism and Bi-LSTM. Then, these two branches were fused by a simple operation (Tijera and Ochoa-Luna, 2018). Our best result on this set were obtained when we use element-wise product as fusing operation. Besides, this model improve previous experiment sets on both metrics. This model inspired the next set of experiments.

In the last set of experiments, we fused these two branches with a Highway-style gate. this second gate is based on question-summary and previous word representations of branches. These models outperform results of previous experiment sets. In Table 5.7, we can see the impact of question-summary and previous word representations.

Model (Single)	**F1**	**EM**
Logistic Regression (Rajpurkar et al., 2016)	51.0	40.0
Dynamic Chunk Reader (Yu et al., 2016)	71.2	62.5
FG fine-grained gate (Yang et al., 2017)	71.3	60.0
DCN (Xiong et al., 2017)	75.6	65.4
Multi-Perspective Matching (Wang et al., 2016)	75.8	66.1
Match-LSTM with Ans-Ptr (Wang and Jiang, 2017)	77.2	67.0
BiDAF (Seo et al., 2017)	77.3	67.7
BiDAF+SEDT (Liu et al., 2017)	77.6	68.1
BiDAF+Gated branch (×) (Tijera and Ochoa-Luna, 2018)	78.3	68.7
BiDAF+Gate+LSTM+Gate **(Ours)**	78.4	69.1
FastQAExt (Weissenborn et al., 2017)	78.5	70.3
Document Reader (Chen et al., 2017)	78.8	69.5
Ruminating Reader (Gong and Bowman, 2018)	79.5	70.6
R-NET (Wang et al., 2017)	79.5	71.1
MEMEN (Pan et al., 2017)	80.4	71.0
BiDAF+Self-attention (Clark and Gardner, 2018)	80.8	71.6
M-Reader+RL (Hu et al., 2018)	81.6	72.1
DCN+ (Xiong et al., 2018)	83.1	74.5
QANet (Yu et al., 2018)	83.8	75.1
RaSoR+TR+LM(L1) (Salant and Berant, 2018)	84.0	**77.0**
SAN (Liu et al., 2018)	84.1	76.2
FusionNet (Huang et al., 2018)	**85.6**	75.3

Table 5.11: Scoreboard of published result on SQuAD v1.1 development set.

When we ablated the question-summary the model had better mean results. However, in Table 5.10, we noticed that models with question-summary had better results individually, reaching 78.36% in F1 score and 69.1% in exact match.

Table 5.11 shows the scoreboard of related work results, the results was obtained testing the models on SQuAD v1.1 development set. As you can notice, the differences between the results of the models are relatively small, coinciding with improvements shown by Deep Learning (DL) models in other fields.

Chapter 6

Conclusions and Future Work

Currently, Deep Learning (DL) is becoming the state-of-the-art in many research fields, where Question Answering (QA) is not the exception. Modern QA is approached by a reading comprehension style. In this context, Machine Comprehension (MC) models powered by DL are leading all score boards. However, a lot of them are focused on improving the Interaction Encoder (IE), frequently based on attention mechanism. We explored in depth the Context Encoder (CE) architectures, finding many improving possibilities.

Our work proposed the including of gating mechanism into CE. It allowed us to control the flow of information from Document CE toward IE. Our experiments performed on Stanford Question Answering Dataset (SQuAD) benchmark, gave us satisfactory results. Our gated Document CE outperformed the baseline model, reaching 78.36% on F1 score and 69.1% on exact match metric. Exceeding reported baseline results around by 1.1% and 1.4%, on F1 score and exact match, respectively.

Experiments showed the impact of 2-layer Highway network in CE. In our last experiments, when we ablated it from Document CE, the accuracy increased until 0.52% on F1 measure and 0.47% on exact match. Surprisingly, when we ablated it from both CEs, the accuracy increased until 0.23% on F1 score, it just happened in last experiments.

Including of previous Document representation in gating mechanism, had a significant impact. When this feature is ablated, the accuracy drops until 0.26% on F1 score and 0.38% on exact match. On the other hand, when we ablated Question-summary, the accuracy drops around 0.05% in both metrics. Apparently, Question-summary just adds variability to the model, because in mean results its presence decreased the accuracy.

Finally, the impact of second gate over simple element-wise operation, reached a 0.29% on F1 score and 0.3% on exact match, validating its importance in our proposal.

6.1 Limitations

Our work is performed on SQuAD benchmark, considered the ImageNet for MC. However, its number of examples is not enough, requiring some kind of data augmentation method (Yu et al., 2018), or transfer learning by some pre-trained contextual encoding (Salant and Berant, 2018).

6.2 Recommendations

Our work is focusing to improve the CE, However, it is important to notice the affinity between components. As it happened to us with the 2-layer Highway network, which in the first experiments improved the results, and in the last ones combined with our proposal, they made our results worse.

6.3 Future Work

Due to the nature of its aim, our CE can be generalized to other Natural Language Processing (NLP) tasks, such as: Machine Translation (MT), Sentiment Analysis y Text Summarization. Our CE can be included as the first processing layer in several DL models.

Our future work definitively must be include a strong pre-trained Language Model (Salant and Berant, 2018; Peters et al., 2018), given the limitations about SQuAD benchmark and the good results shown by this learning transfer technique. Also, we consider that self-attention after attention mechanism would be a great idea. Self-attention showed an generous increasing on models accuracy (Clark and Gardner, 2018; Wang et al., 2017).

Besides, consider a training process based on Reinforcement learning (RL), few works deal with it. However, their results are promising (Xiong et al., 2018; Hu et al., 2018). Answer Decoder (AD) also has few related work, proposing architectures for this could mean a great contribution (Liu et al., 2018).

Bibliography

Anderson, J. (2005). *Cognitive Psychology and Its Implications*. Worth Publishers.

Bahdanau, D., Cho, K., et al. (2015). Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Bengio, Y., Simard, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Berant, J., Srikumar, V., et al. (2014). Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1510. Association for Computational Linguistics.

Britz, D., Goldie, A., et al. (2017). Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451. Association for Computational Linguistics.

Chan, W., Jaitly, N., et al. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

Chelba, C., Mikolov, T., et al. (2013). One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Chen, D., Bolton, J., et al. (2016). A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 2358–2367. Association for Computational Linguistics.

Chen, D., Fisch, A., et al. (2017). Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 1870–1879. Association for Computational Linguistics.

Cheng, J., Dong, L., et al. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., et al. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Clark, C. and Gardner, M. (2018). Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 845–855. Association for Computational Linguistics.

Cui, Y., Chen, Z., et al. (2017). Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 593–602. Association for Computational Linguistics.

Dang, H. T., Kelly, D., et al. (2007). Overview of the TREC 2007 question answering track. In *Proceedings of the Sixteenth Text REtrieval Conference*, volume 1, pages 105–122. NIST Publications.

Dhingra, B., Liu, H., et al. (2017). Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 1832–1846. Association for Computational Linguistics.

Ferrucci, D., Brown, E., et al. (2010). Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Gong, Y. and Bowman, S. (2018). Ruminating reader: Reasoning with gated multi-hop attention. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 1–11. Association for Computational Linguistics.

Goodfellow, I., Bengio, Y., et al. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Goodfellow, I. J., Warde-Farley, D., et al. (2013). Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1319–III–1327. JMLR.org.

Grabe, W. (2009). *Reading in a Second Language: Moving from Theory to Practice*. Cambridge Applied Linguistics. Cambridge University Press.

Green, Jr., B. F., Wolf, A. K., et al. (1961). Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), pages 219–224, New York, NY, USA. Association for Computing Machinery.

Hermann, K. M., Kočiský, T., et al. (2015). Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Hill, F., Bordes, A., et al. (2015). The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91:1.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hu, M., Peng, Y., et al. (2018). Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Huang, H.-Y., Zhu, C., et al. (2018). Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations*.

Joshi, M., Choi, E., et al. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551.

Józefowicz, R., Vinyals, O., et al. (2016). Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*, volume 3. Pearson London.

Kadlec, R., Schmid, M., et al. (2016). Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 908–918. Association for Computational Linguistics.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kolomiyets, O. and Moens, M.-F. (2011). A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412 – 5434.

Ling, W., Tsvetkov, Y., et al. (2015). Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1367–1372. Association for Computational Linguistics.

Liu, R., Hu, J., et al. (2017). Structural embedding of syntactic trees for machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 826–835. Association for Computational Linguistics.

Liu, X., Shen, Y., et al. (2018). Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 1694–1704. Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., et al. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

McCann, B., Bradbury, J., et al. (2017). Learned in translation: Contextualized word vectors. In Guyon, I., Luxburg, U. V., et al., editors, *Advances in Neural Information Processing Systems 30*, pages 6294–6305. Curran Associates, Inc.

Onishi, T., Wang, H., et al. (2016). Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235. Association for Computational Linguistics.

Pan, B., Li, H., et al. (2017). MEMEN: multi-layer embedding with memory networks for machine comprehension. *CoRR*, abs/1707.09098.

Paperno, D., Kruszewski, G., et al. (2016). The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 1525–1534. Association for Computational Linguistics.

Parikh, A., Täckström, O., et al. (2016). A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255. Association for Computational Linguistics.

Pennington, J., Socher, R., et al. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics.

Peters, M., Neumann, M., et al. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Rajpurkar, P., Zhang, J., et al. (2016). Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Richardson, M., Burges, C. J., et al. (2013). Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203. Association for Computational Linguistics.

Rocktäschel, T., Grefenstette, E., et al. (2016). Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Salant, S. and Berant, J. (2018). Contextualized word representations for reading comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 554–559. Association for Computational Linguistics.

Seo, M. J., Kembhavi, A., et al. (2017). Bi-directional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Shen, Y., Huang, P.-S., et al. (2017a). Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1047–1055, New York, NY, USA. Association for Computing Machinery.

Shen, Y., Liu, X., et al. (2017b). An empirical analysis of multiple-turn reasoning strategies in reading comprehension tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 1 (Long Papers), pages 957–966. Asian Federation of Natural Language Processing.

Sordoni, A., Bachman, P., et al. (2016). Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245.

Srivastava, N., Hinton, G., et al. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Srivastava, R. K., Greff, K., et al. (2015a). Highway networks. In *Deep Learning Workshop at International Conference on Machine Learning*.

Srivastava, R. K., Greff, K., et al. (2015b). Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 2377–2385, Cambridge, MA, USA. MIT Press.

Tijera, C. M. and Ochoa-Luna, J. (2018). Fine-grained gating based on question-summary for machine comprehension. In *17th Mexican International Conference on Artificial Intelligence*.

Trischler, A., Ye, Z., et al. (2016). Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 128–137. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., et al. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., et al., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Vinyals, O., Fortunato, M., et al. (2015). Pointer networks. In Cortes, C., Lawrence, N. D., et al., editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Voorhees, E. M., Harman, D. K., et al. (2005). *TREC: Experiment and Evaluation in Information Retrieval*, volume 1. MIT press Cambridge.

Wang, S. and Jiang, J. (2016). Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451. Association for Computational Linguistics.

Wang, S. and Jiang, J. (2017). Machine comprehension using match-lstm and answer pointer. In *International Conference on Learning Representations*.

Wang, W., Yang, N., et al. (2017). Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1 (Long Papers), pages 189–198. Association for Computational Linguistics.

Wang, Z., Mi, H., et al. (2016). Multi-perspective context matching for machine comprehension. *CoRR*, abs/1612.04211.

Weissenborn, D., Wiese, G., et al. (2017). Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280. Association for Computational Linguistics.

Weizenbaum, J. (1966). Eliza–a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

Woods, W. A. (1978). Semantics and quantification in natural language question answering. volume 17 of *Advances in Computers*, pages 1–87. Elsevier.

Xiong, C., Zhong, V., et al. (2017). Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.

Xiong, C., Zhong, V., et al. (2018). DCN+: Mixed objective and deep residual coattention for question answering. In *International Conference on Learning Representations*.

Yang, Y., Yih, W.-t., et al. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018. Association for Computational Linguistics.

Yang, Z., Dhingra, B., et al. (2017). Words or characters? fine-grained gating for reading comprehension. In *International Conference on Learning Representations*.

Yu, A. W., Dohan, D., et al. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

Yu, Y., Zhang, W., et al. (2016). End-to-end answer chunk extraction and ranking for reading comprehension. *CoRR*, abs/1610.09996.

Zhang, J., Zhu, X., et al. (2017). Exploring question representation and adaptation with neural networks. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 1975–1984.

Appendix A

Bi-Directional Attention Flow

A.1 BiDAF Overview

An overview of the baseline model is presented in Figure A.1.

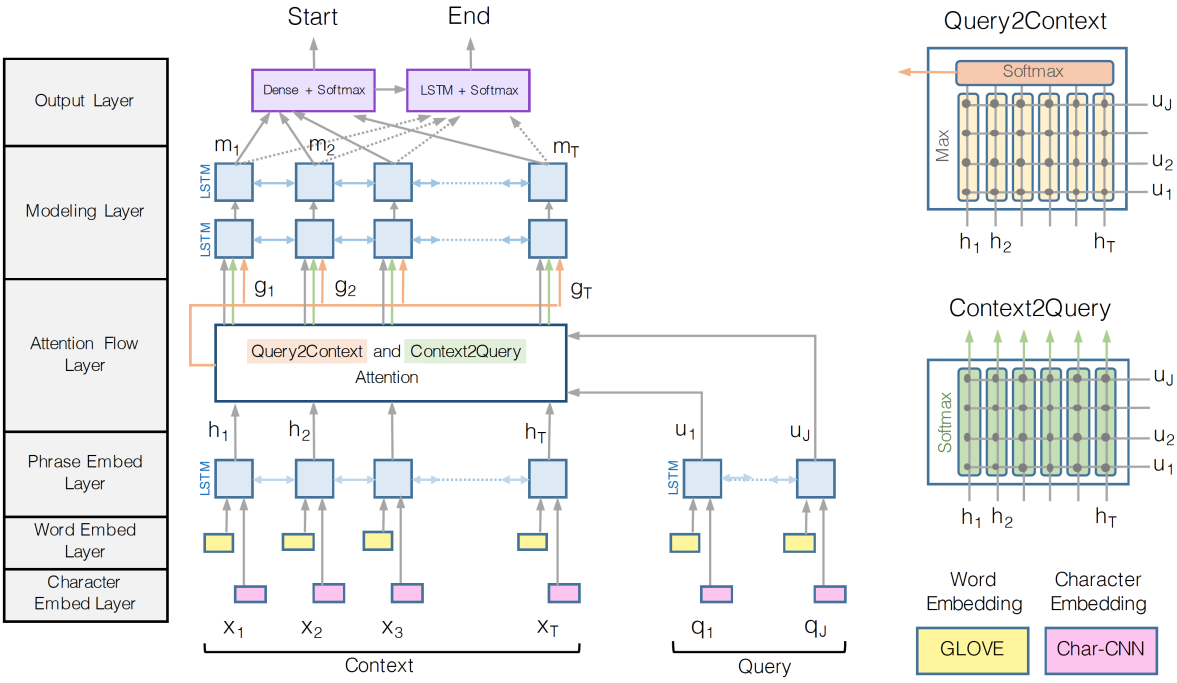


Figure A.1: Overview of BiDAF model extracted from (Seo et al., 2017).

A.2 Error Analysis

Table A.1 summarizes the modes of errors by Bi-Directional Attention Flow (BiDAF) and shows examples for each category of error in SQuAD.

Error type	%	Example
Imprecise answer boundaries	50	**Context:** "The Free Movement of Workers Regulation articles 1 to 7 set out the main provisions on equal treatment of workers." **Question:** "Which articles of the Free Movement of Workers Regulation set out the primary provisions on equal treatment of workers?" **Prediction:** "1 to 7", **Answer:** "articles 1 to 7"
Syntactic complications and ambiguity	28	**Context:** "A piece of paper was later found on which Luther had written his last statement." **Question:** "What was later discovered written by Luther?" **Prediction:** "A piece of paper", **Answer:** "his last statement"
Paraphrase problems	14	**Context:** "Generally, education in Australia follows the three-tier model which includes primary education (primary schools), followed by secondary education (secondary schools/high schools)..." **Question:** "What is the first model of education, in the Australian system?" **Prediction:** "three-tier", **Answer:** "primary education"
External knowledge	4	**Context:** "NFL announced that the practice of branding Super Bowl games with Roman numerals, a practice established at Super Bowl V, would be temporarily suspended, and that the game would be named using Arabic numerals as Super Bowl 50 as opposed to Super Bowl L." **Question:** "If Roman numerals were used in the naming of the 50th Super Bowl, which one would have been used?" **Prediction:** "Super Bowl 50", **Answer:** "L"
Multi-sentence	2	**Context:** "Over the next several years in addition to host to host interactive connections the network was enhanced to support terminal to host connections, host to host batch connections (remote job submission, remote printing, batch file transfer), interactive file transfer, gateways to the Tymnet and Telenet public data networks, X.25 host attachments, gateways to X.25 data networks, Ethernet attached hosts, and eventually TCP/IP and additional public universities in Michigan join the network. All of this set the stage for Merit's role in the NSFNET project starting in the mid-1980s." **Question:** "What set the stage for Merits role in NSFNET" **Prediction:** "All of this set the stage for Merit's role in the NSFNET project starting in the mid-1980s", **Answer:** "Ethernet attached hosts, and eventually TCP/IP and additional public universities in Michigan join the network"
Incorrect preprocessing	2	**Context:** "English chemist John Mayow (1641-1679) refined this work by showing that fire requires only a part of air that he called spiritus nitroaereus or just nitroaereus." **Question:** "John Mayow died in what year?" **Prediction:** "1641-1679", **Answer:** "1679"

Table A.1: Error analysis on SQuAD v1.1. Seo et al. (2017) randomly selected EM-incorrect answers and classified them into 6 different categories. Only relevant sentence(s) from the context shown for brevity.